PROJETO DE UMA UNIDADE DE CONTROLE MICROPROGRAMADA PARA UMA UCP DE MÉDIO PORTE

Adriano Joaquim de Oliveira Cruz

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.).

Aprovada por:

Ysmar Vianna e Silva Filho

YSMAR VIANNA E SILVA FILHO
Presidente

Nelson Maculan Filho

NELSON MACULAN FILHO

Ivan da Costa Marques

IVAN DA COSTA MARQUES

RIO DE JANEIRO - RJ - BRASIL

NOVEMBRO DE 1979

CRUZ, ADRIANO JOAQUIM DE OLIVEIRA

Projeto de uma Unidade de Controle Microprogramada para uma UCP de Médio Porte (RIO DE JANEIRO) 1979.

VII, 85 p., 29,7 cm (COPPE - UFRJ, M.Sc, Engenharia de Sistemas, 1975)

Tese - Universidade Federal do Rio de Janeiro - COPPE - UFRJ.

1. Microprogramação I. COPPE/UFRJ II Título (série).

A G R A D E C I M E N T O

Aos colegas ADALBERTO AFONSO BARBOSA,AGEU CAVALCANTI PACHEÇO JÚNIOR, ARMANDO DRUMMOND, JULIO SALEK AUDE, MARIO FERREIRA MARTINS, PAULO HENRIQUE OE AGUIAR RODRIGUES e ROGÉRIO ANTONIO SAMPAIO PARENTE VIANNA pelo auxílio técnico e apoio que deram durante o desenvolvimento do projeto.

A JAYME LUIZ SZWARCFITER e YSMAR VIANNA E SILVA FILHO pelo apoio dado ao projeto.

E finalmente aos estagiários ANTONIO CEZAR PEREIRA DE MENDONÇA UCHÔA, CARLO EMMANUEL TOLLA DE OLIVEIRA, CYBELE LUZANA REIS e JOSÉ RICARDO PEREIRA RIBEIRO pelo trabalho desenvolvido na fase de montagem de projeto.

R E S U M O

Este trabalho descreve uma Unidade de Controle projetada para um processador de médio porte atualmente em desenvolvimento no NCE/UFRJ.

Trata-se de uma Unidade de Controle microprogramada, com microprogramação fixa armazenada em memórias de leitura exclusiva, sendo a palavra de controle do tipo horizontal.

Inicialmente é apresentada uma visão geral do processador, seguida de uma descrição da Unidade Aritmética com o objetivo de mostrar os recursos que o microprogramador dispõe para desenvolver as micro-rotinas das instruções inteiras e de ponto flutuante.

O restante do trabalho é dedicado a apresentação e discussão dos circuitos da Unidade de Controle e dos microprogramas das instruções inteiras.

ABSTRACT

This work presents a Control Unit design for a medium size processor under development at NCE/UFRJ.

Fixed microprogramming approach has been selected, using horizontal type control words.

A general survey about the processor is initially presented, followed by an Arithmetic Unit description in order to show the programmer's available features for integer and floating-point micro-routines implementation.

The last part presents and discusses Control Unit hardware and integer instructions microprograms.

Í N D I C E

I - Introdução

Este trabalho apresenta o projeto de uma Unidade de Controle para uma Unidade Central de Processamento (UCP) de médio porte, compatível em "software" com o computador PDP 11/70 da Digital Equipment Corporation (D.E.C.), que vem sendo desenvolvida no Núcleo de Computação Eletrônica da U.F.R.J. (NCE).

A idéia de se construir uma UCP de médio porte surgiu da necessidade que o NCE sentia de contribuir para o esforço que algumas Universidades e instituições faziam e continuam a fazer para criar no país tanto a tecnologia necessária para o projeto e construção de equipamentos digitais como pessoal qualificado a manter e ampliar este esforço.

A escolha de um computador de médio porte era justificada pelo estágio em que se encontrava na época o conhecimento sobre projetos digitais no Brasil. A experiência acumulada pelas Universidades havia resultado no mini-computador G-10, projeto conjunto da USP e PUC-RJ, que estava concluído e pronto para ser industrializado. Deste modo uma UCP de médio porte seria um passo à frente natural dado pelos técnicos objetivando ampliar a fronteira dos conhecimentos até uma faixa superior de computadores.

As iniciativas para a industrialização do G-10 resultaram infrutíferas por diversos motivos. No plano técnico havia o argumento de que devido ao fato do computador ter um "software" novo, era pouco conhecido dos usuários e insuficientemente testado. Este argumento que em parte era verdadeiro estabelece um círculo vicioso; não se usa o computador porque seu "software" não foi testado, não são feitos testes definitivos porque não é nem será usado, daí não se constrói nada. Para romper este esquema é necessário o estabelecimento de uma política de desenvolvimento que esteja voltada para interesses nacionais dando apoio a tecnologia produzida no país. Entretanto estas iniciativas não foram de todo frustradas, pois uma das consequências das discussões iniciadas com a industrialização do G-10 foi a resolução da CAPRE que reservava para a indústria nacional a faixa dos mini-computadores.

Deste argumento usado contra o G-10, ressalvando como já mostramos que é possível contorná-lo, tiramos um dos motivos que nos levaram a projetar um computador compatível em

"software" com outro já testado e de amplo conhecimento dos usuários. Assim sua industrialização poderá ser feita sem os supostos inconvenientes que um "software" totalmente novo traz.

O segundo motivo que nos levou a fazer a opção por uma CPU compatível, está relacionado com uma limitação do próprio NCE. Não havia na instituição pessoal suficiente para desenvolver e testar em curto prazo um "software" integralmente original.

A necessidade de terminar o projeto em curto prazo tem a sua justificativa na experiência adquirida durante a luta pela implantação de uma indústria nacional de mini-computadores, na qual o G-10 teve grande importância. A natural ampliação desta luta para outras faixas de computadores objetivando abrir espaço para o crescimento das indústrias nacionais, pode ter seu caminho facilitado pela existência de projetos nacionais em funcionamento.

Deste modo a opção de fazer uma UCP "software" compatível pode ser discutida em virtude dos prejuizos evidentes que traz ao desenvolvimeto do "software" básico no Brasil. Entretanto, devido as condições de recursos do NCE e aos fatores externos, esta nos pareceu a maneira mais eficiente que dispúnhamos para contribuir ao esforço de criação de uma indústria nacional de computadores. Neste nacional incluímos os pressupostos que a palavra exige: desenvolvimento de programas e circuitos no país; satisfação de necessidades reais da sociedade brasileira, discutidas em debates abertos e amplos.

II - MICROPROGRAMAÇÃO

1- Introdução

Um considerável avanço nas técnicas de projeto de unidades de controle para computadores digitais, ocorreu quando da idealização da microprogramação. A idéia, que consiste basicamente em substituir os circuitos convencionais anteriormente usados na unidade de controle por uma memória onde é armazenada a lógica de controle do computador (a microprogramação), é normalmente atribuida a Wilkes[1,2]. O principal interesse de Wilkes e sua equipe era organizar os métodos de projeto até então utilizados, antevendo as vantagens que um método ordenado e sistemático traria tanto ao próprio projeto como a sua posterior manutenção. Em artigo publicado em 1958 por Wilkes, Renwick e Weeler[3], há este comentário sobre o problema:"Em muitas máquinas o projeto da unidade de controle tem sido feito por meio de métodos semi-empíricos, e os circuitos resultantes, embora possam ser eficientes, são complexos e não sistemáticos, e poderiam requerer alterações consideráveis se qualquer mudança apreciável tivesse de ser feita no conjunto de instruções da máquina".

Apesar das vantagens técnicas que a microprogramação possui, somente na década de 60 este método de projeto começou a ser aplicado por grandes fabricantes em computadores comerciais. Este crescimento de interesse ocorreu a medida que a microprogramação passou a ser economicamente superior aos projetos com lógica convencional. Esta superioridade econômica se tornou possível entre outras razões pelo aparecimento de memórias de leitura exclusiva (MLE), rápidas e confiáveis para aplicação na indústria.

A microprogramação possibilitou implementar economicamente, por exemplo, um mesmo conjunto de instruções (códigos de operações) para toda uma linha de computadores, mesmo os menores. Desta maneira podemos ter computadores com organizações internas e tecnologias completamente diferentes e que são arquiteturalmente compatíveis. Aqui o termo arquitetura é usado para significar os aspectos do computador que são visíveis ao programador, incluindo então, registros, modos de endereçamento e instruções. Organização, por outro lado se refere a um nível abaixo deste e está significando os registros de trabalho, unidade arit

mética, barras de comunicação de dados, etc, que são usualmente transparentes ao programador.

A microprogramação também tornou possível computadores rodarem com pequenas ou nenhuma alteração, programas de máquinas de linhas anteriores eliminando a necessidade de reprogramação. Estes tipos de compatibilidades, com linhas anteriores e acima e abaixo em uma mesma linha, receberam farta publicidade dos fabricantes junto aos usuários pelas inúmeras vantagens que traziam, demonstrando a importância da microprogramação.Finalmente, também foram conseguidas compatibilidades entre computadores de diferentes fabricantes.

Até aqui fizemos referência a microprogramação somente como uma ferramenta do projetista de computadores. Realmente as primeiras máquinas foram projetadas com o objetivo principal de implementar uma arquitetura particular, não havendo interesse em generalizar o uso da microprogramação levando-a até os usuários. As modificações introduzidas nos microprogramas com a finalidade de emular outras máquinas eram implementadas em MLEs e feitas pelos próprios fabricantes. Entretanto, a introdução de memórias de controle possíveis de serem alteradas,fez com que a microprogramação evoluisse de campo de trabalho dos projetistas de computadores para uma área de interesse de toda comunidade de usuários. Recentemente, começaram a aparecer máquinas que são completamente microprogramáveis e cujas arquiteturas são projetadas para permitir microprogramas de uso geral, podendo então o microprogramador escolher o conjunto de instruções mais apropriado para a sua aplicação.

Como vimos a microprogramação, para surpresa de muitos que a julgavam uma idéia nova está presente de um modo ou de outro, há quase três décadas na área de computação. Há grande quantidade de textos publicados, cobrindo todas as áreas de aplicação da microprogramação e descrevendo as principais idéias e conceitos. Entretanto com o objetivo de estabelecer uma perspectiva comum, este capítulo apresenta os conceitos básicos da microprogramação e algumas definições que serão usadas no restante do trabalho.

2- Estrutura Básica dos Computadores

Um computador digital moderno e em geral qualquer sistema que trate informação, pode ser dividido em quatro unidades com atividades distintas: Unidade Aritmética (UA), Unidade de Entrada e Saída (UE/S), Unidade de Controle (UC) e Memória. Estas unidades comunicam-se entre si, trocando dados, instruções e controles como está representado na figura II-1. Os dois primeiros tipos de informação fazem parte da rotina de trabalho de todos que usam computadores, enquanto que os sinais de controle são quase sempre familiares somente aos técnicos que trabalham ao nível dos circuitos da máquina.

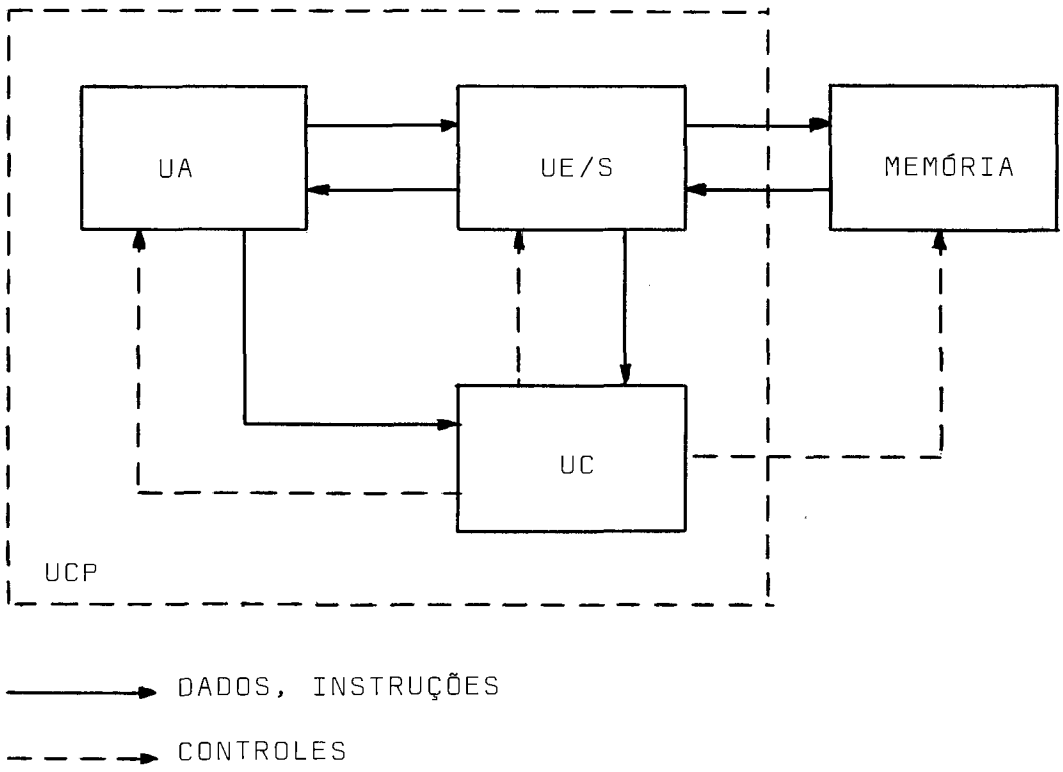


fig.II-1 - Elementos de um Computador Digital.

O que é normalmente chamado de Unidade Central de Processamento (UCP) do computador é composto da UA, UE/S e UC. As duas primeiras unidades estão relacionadas às transformações e movimentações de dados que ocorrem durante o processamento e a última com o controle da UCP.

3- Unidade Aritmética

A Unidade Aritmética é constituida na maioria das máquinas de elementos armazenadores, circuitos que realizam operações e caminhos de dados. Os elementos armazenadores incluem registros para operandos e endereços, memórias e elementos que armazenam condições para testes (ocorrência de transbordo, pedidos de interrupção, etc). Dos circuitos que realizam operações o mais importante e o coração da UA, é a chamada Unidade Aritmética e Lógica (UAL). Na UAL são executadas as operações básicas, que correspondem ao repertório de funções da UCP ao nível de circuitos. Este repertório consiste de operações como, SOMA, SUBTRAÇÃO, E, OU, COMPLEMENTAÇÃO, etc, e a partir destas podem ser sintetizadas operações mais complexas. Normalmente também há circuitos que realizam operações especializadas como contagem, comparações de endereços e dados. O restante da UA é constituido de multiplexadores, caminhos de dados e portas lógicas, que servem para interconectar os circuitos armazenadores e os que realizam as operações.

A figura II-2 mostra uma UA simples. Neste exemplo, a entrada A da UAL pode receber os registros A ou B através do caminho A; a entrada B, os registros C ou D através do caminho B; a saída UAL pode ser carregada em qualquer dos registros através do caminho T; e assumiremos que a UAL pode executar diversas funções básicas. É evidente que para a UA realizar uma

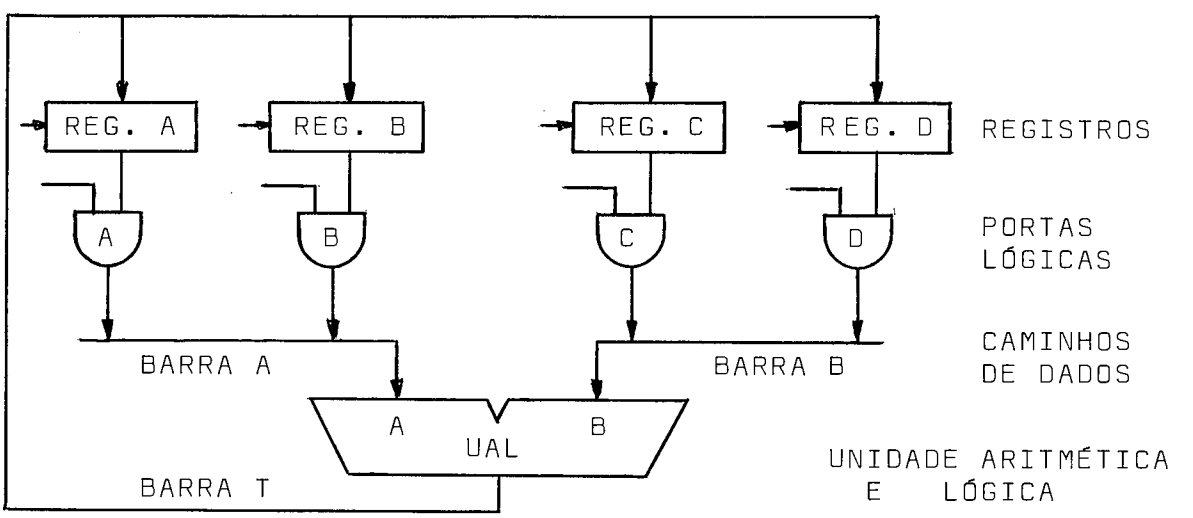


fig. II-2 - Unidade Aritmética

operação qualquer, por exemplo somar os conteúdos dos registros A e C, sinais de controle devem ser aplicados as portas lógicas A e C de modo que os dados sejam liberados nos caminhos apropriados e programar a UAL para realizar a operação selecionada. Como é necessário esperar um espaço de tempo para que os dados atravessem as portas, caminhos e que a operação seja realizada, os sinais devem permanecer ativos este tempo. Geralmente é escolhido o tempo que é gasto na operação mais demorada através da UA como o ciclo da máquina. Operações mais complexas são realizadas com um número inteiro destes ciclos. O estado em que está o conjunto dos sinais de controle durante um ciclo, definem o que se costuma chamar de estado da máquina.

4- Unidade de Controle

Todos os sinais que controlam a operação da UA e de toda UCP, provem da Unidade de Controle. Para emitir estes sinais, a UC recebe informações da Memória que especificam que operação deve ser realizada e a localização dos dados que serão operados. De posse destas informações a UC determina (decodifica) exatamente qual é a instrução, comanda as movimentações e transformações com os endereços e operandos e ao final da execução dá início ao processo de busca de uma nova instrução. Cada instrução é então um pedido a UC para ativar os circuitos do sistema que realizam as operações lógicas ou aritméticas e abrir e fechar certas portas lógicas que permitem os dados caminhar pelas diversas unidades, isto durante um número pré determinado de ciclos. Podemos afirmar que as tarefas da UC são de dois tipos: definir os sinais que serão ativados durante um ciclo da máquina e determinar qual será o próximo estado.

Em uma UC de controle convencional estas tarefas são realizadas por um conjunto de circuitos lógicos combinacionais e sequenciais, espalhados pelo processador, que funcionam como uma máquina de número finito de estados. Como vimos este tipo de UC tem um método de projeto não sistemático, que traz problemas durante a própria fase de projeto e depois na manutenção. Wilkes[1] propôs então uma alternativa que visava eliminar estas dificuldades.

Como já observamos, a execução de uma instrução é dividida em uma série de ciclos, onde são realizadas operações básicas (chamadas micro-ordens), comandadas por sinais oriundos da UC. Tais micro-ordens podem por exemplo comandar uma transferência entre registros, ordenar uma soma, etc, por meio da abertura e fechamento de certas portas lógicas. A idéia básica de Wilkes era controlar estas portas, registros, etc. através de linhas de controle, cujo estado podendo ser representado por zero ou um, seria armazenado em uma matriz. O esquema original de Wilkes está mostrado na figura II-3. Ele consiste de uma matriz

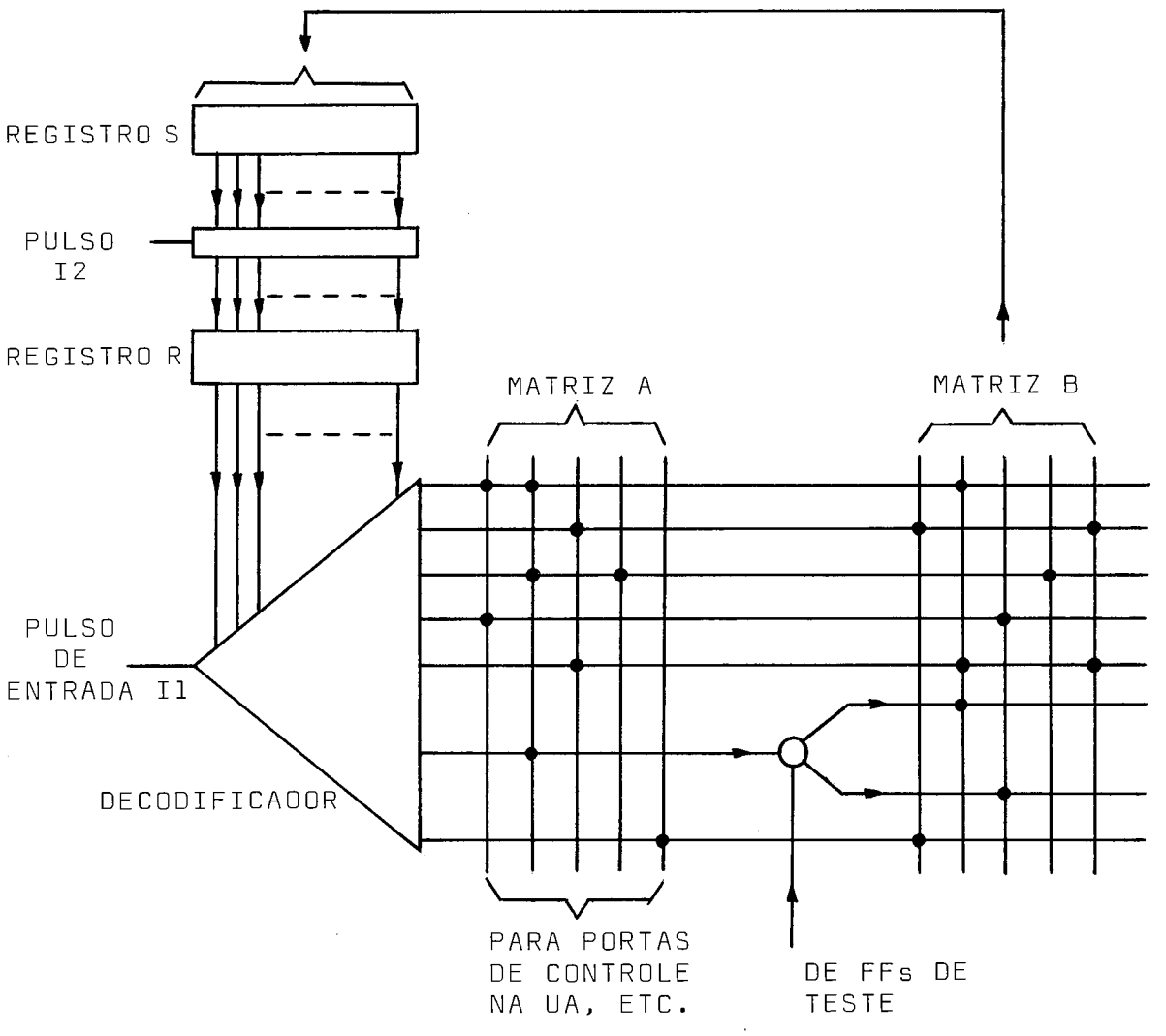


fig.II-3 - Unidade de Controle no Modelo de Wilkes

de controle A, outra de sequenciamento B, circuitos de sincronis

mo e um decodificador (foram originalmente sugeridas matrizes de diodos, embora Wilkes tenha apontado outras soluções[3]). Na matriz A as linhas verticais são conectadas a uma única porta de controle no sistema, podendo então estas portas serem abertas ou fechadas simplesmente ativando-se a linha de controle correspondente na matriz A. O circuito decodificador recebe a saída do registro R, decodifica seus n bits e seleciona uma entre 2^n linhas horizontais para ser ativada, isto é, em cada ciclo o decodificador permite que seja ativada somente uma linha. A linha é ativada por meio de um pulso de sincronismo (Il) vindo do decodificador. Ela por sua vez, de acordo com a disposição dos diodos na matriz, ativa um conjunto de portas lógicas no sistema, fazendo com que sejam realizadas as operações elementares apropriadas. A linha horizontal pode ser entendida como uma instrução da Unidade de Controle ou uma microinstrução.

O pulso de sincronismo passa também através da matriz B, e a sua saída é armazenada no registro S. No começo de um novo ciclo o conteúdo de S é transferido para o registro R, que apontará então a nova microinstrução a ser executada.

Há uma certa analogia na maneira como a máquina executa os diversos passos requeridos para implementar uma instrução e a maneira com que são executadas as diversas instruções que constituem um programa. Isto sugeriu o termo microprogramação como definição do processo de desenvolver um conjunto de microinstruções (o conteúdo da matriz de controle), utilizadas para controlar as operações dos diversos elementos dentro da UCP, de modo que seja executado o conjunto de instruções da máquina.

Uma outra idéia sugerida por Wilkes e Stringer[4] refere-se a necessidade de haver desvios condicionais nos microprogramas. A sugestão era fazer com que o endereço da próxima microinstrução a ser executada, fosse função de um flip-flop que armazenaria a condição a ser testada. Dependendo do estado deste flip-flop, um de dois endereços é selecionado (ver fig II-3), podendo-se deste modo testar por exemplo, bits de um multiplicador durante a microprogramação de uma instrução de multiplicação. Ainda de Wilkes, foi a sugestão de fazer com que o conteúdo do registro S, pudesse ser carregado, ou com a saída da matriz B ou a partir outra fonte, que poderia ser o próprio código de operação da instrução. Deste modo seria eliminada a necessidade de um cir

cuito decodificador de instruções. Também foi especulado que poderia haver módulos de matrizes intercambiáveis, cada um com microprogramação para conjuntos de instruções diferentes. Deste modo cada usuário poderia escolher o conjunto de instruções mais apropriado para sua aplicação. Além dessas modificações, outros esquemas foram sugeridos por Wilkes e sua equipe.

Existem vários parâmetros que são utilizados para descrever as microinstruções. Apesar de não haver definições precisas sobre os seus significados eles contém informações significativas sobre as microinstruções. Geralmente são usados para descrever as microinstruções o seu comprimento, a quantidade de informação contida na palavra e o seu grau de codificação, a organização da informação dentro da microinstrução e a sincronização da execução. Todos esses parâmetros serão apresentados mais adiante durante a descrição do projeto da Unidade de Controle, junto com o seu enquadramento dentro destes parâmetros.

III - DESCRIÇÃO GERAL DO SISTEMA

1 - Introdução

Com a finalidade de apresentar uma visão geral da arquitetura do sistema e da organização interna da UCP, este capítulo faz uma breve apresentação do projeto como um todo, mostrando suas características mais importantes e destacando os aspectos principais exitentes nos projetos da Unidade Aritmética, Unidade de Entrada e Saída, Sistema de Memória e Painel.

O fato de se ter como proposta a realização do projeto de uma UCP compatível em "software" com o PDP-11/70 da D.E.C. não implicou, de forma alguma, na adoção de uma filosofia de trabalho que não considerasse importante a busca de soluções novas e criativas para a execução do projeto. Apenas no plano de definição da máquina, do ponto de vista do programador é que poucas alterações puderam ser feitas. No entanto sua organização interna, isto é, os circuitos da Unidade de Controle, Unidade Aritmética, Unidade de Entrada e Saída e Sistema de Memória são completamente diferentes, resultado de um novo projeto.

A exigência de compatibilidade em "software" com o PDP 11/70 determinou duas diretrizes para o projeto. A primeira delas é que a UCP a ser projetada fosse capaz de executar todo o conjunto de instruções do PDP-11/70 e que a execução de qualquer instrução do conjunto provocasse, do ponto de vista do programador, efeitos idênticos nesta UCP e no PDP 11/70. A segunda obriga que a UCP opere com os registros internos, indicadores de erro, "status" e condições de operação do sistema, de forma idêntica ao PDP-11/70

Uma terceira diretriz para o projeto foi definida pelo fato de que seria mais econômico e prático permitir que a esta UCP pudessem ser conectados todos os periféricos da D.E.C, sem a necessidade de utilização de circuitos especiais. A adoção desta idéia traz como consequência a utilização de um protocolo de troca de sinais com periféricos idêntico ao empregado no PDP-11/70.

A obediência a estas diretrizes durante o projeto, permitirá a utilização da UCP sob controle de Sistemas Operacionais desenvolvidos pela D.E.C para computadores da linha PDP-11/70, bem como o acoplamento a ela de uma grande variedade de

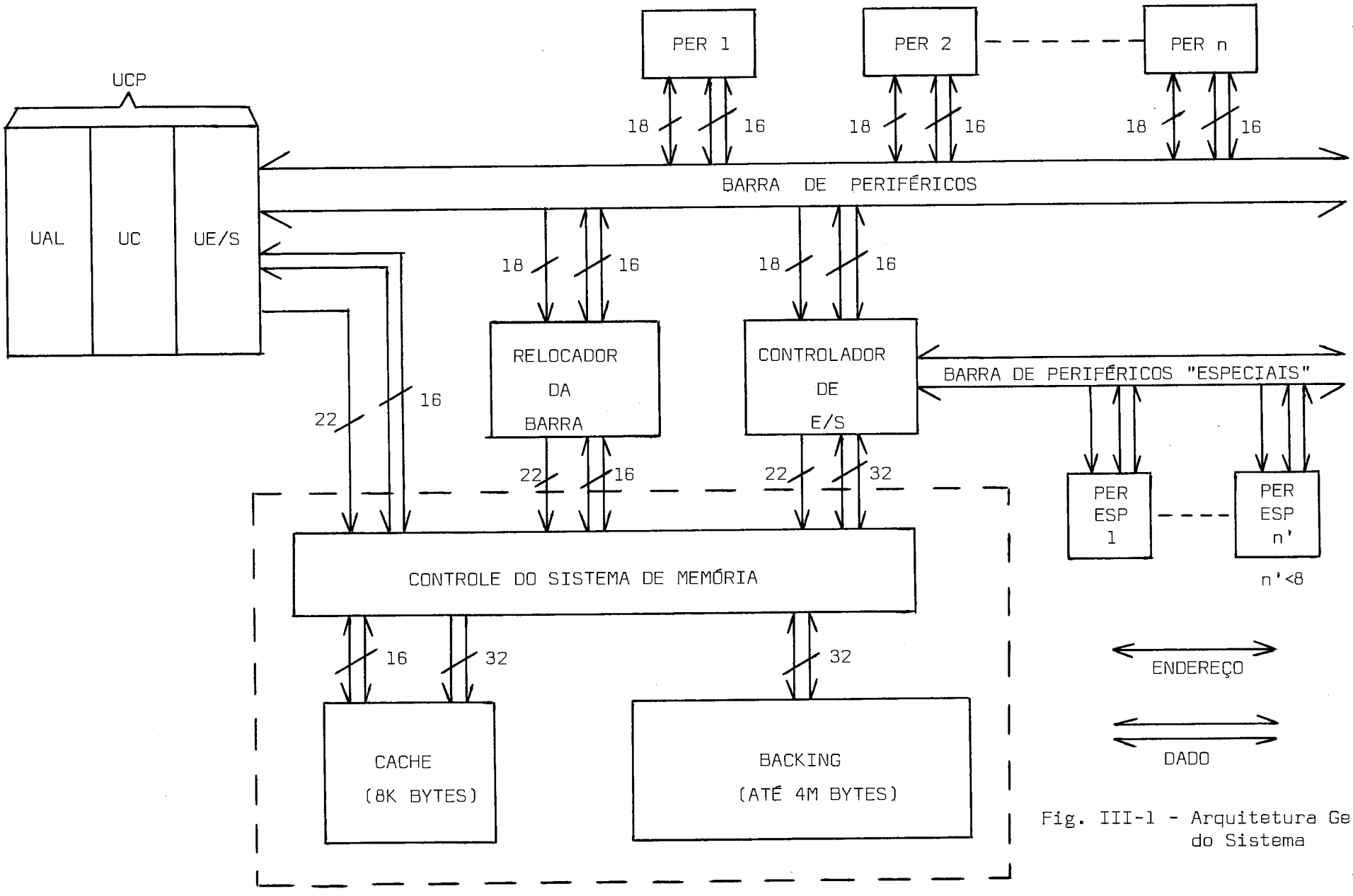


Fig. III-1 - Arquitetura Geral do Sistema

periféricos.

2- Características Gerais do Processador

2.1- Barras de Comunicação

A arquitetura geral do sistema é mostrada de forma simplificada na fig.III-1.O processador constituido pela Unidade Aritmética, Unidade de Controle e Sistema de Entrada e Saída, se comunica com os periféricos e com o Sistema de Memória através de barras de comunicação que operam independentemente. O Sistema de Memória pode ainda ser acessado por periféricos ligados a Barra de Periféricos através de um Sistema de Relocação de Endereços, dispositivo que transforma os endereços de 18 bits gerados pelos periféricos em endereços de memória que são de 22 bits. Finalmente temos os Controladores de E/S, que comandam periféricos de alta velocidade (discos e fitas magnéticas de grande capacidade) em operações de acesso direto a memória.

2.2- Modos de Operação

O processador possui três modos de operação: Kernel, Supervisor e Usuário, sendo o modo Kernel o mais privilegiado. Neste modo um programa tem controle completo da máquina, podendo ser executada qualquer instrução. Quando a máquina está operando em um dos outros modos, certas instruções não são executadas, podendo também ser negado acesso direto a periféricos do sistema. O Sistema Operacional executa em modo Kernel as rotinas de tratamento de erro, rotinas de entrada e saída, gerência de memória, etc. No modo Supervisor são executados programas que tratam arquivos do sistema e os compiladores, editores, montadores. A indicação do modo corrente de operação do processador é dada por dois bits da Palavra de Estado do Processador, que possui ainda outros dois bits indicadores do modo anterior do processador.

2.3- Registros Referenciáveis por Instrução

No conjunto de registros referenciáveis por ins

trução temos os chamados registros gerais, mostrados na figura III-2, que são 16 ao todo: RØ a R5 (conjuntos Ø e 1), R6 (Kernel, Supervisor e Usuário) e o R7. Estes registros podem ser usados como acumuladores, apontadores indexadores e apontadores incrementados ou decrementados automaticamente a cada acesso à memória.

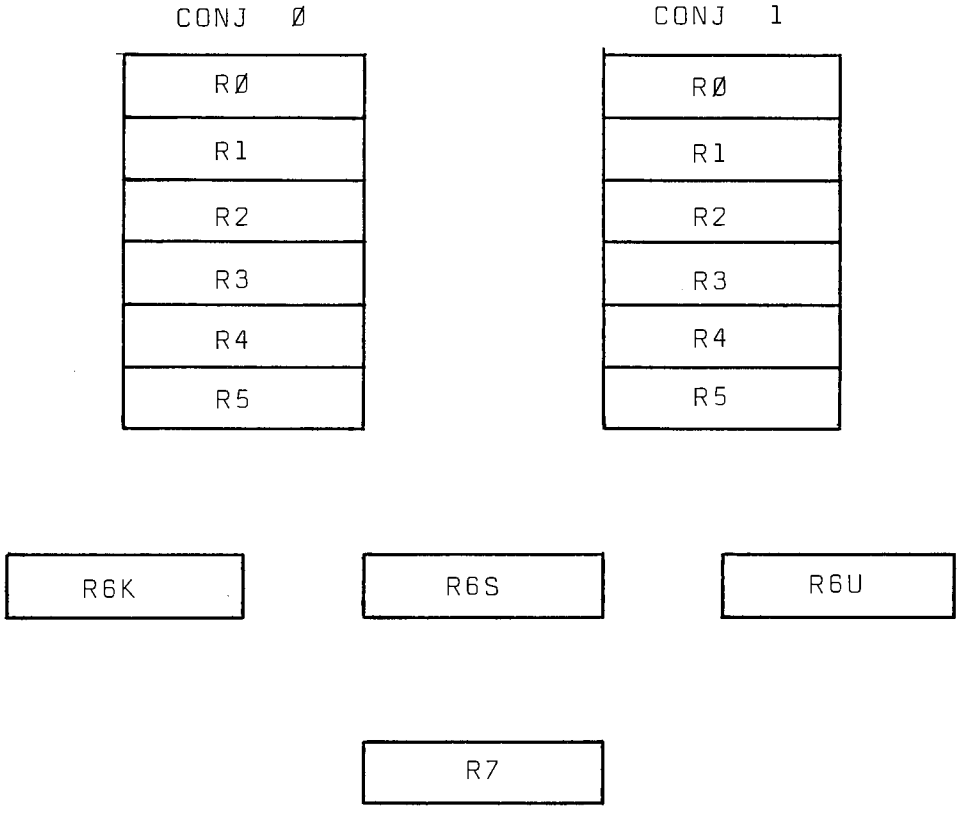


Figura III-2 - Registros Gerais

A existência de dois conjuntos de registros gerais RØ a R5, tem por objetivo tornar mais rápido o procedimento de tratamento de interrupções pela UCP. Normalmente um dado programa utiliza-se de um dos conjuntos de registros gerais, sendo o outro utilizado pelas rotinas de tratamento de interrupção. Deste modo a troca de rotinas durante as interrupções é mais rápida, já que não é necessário salvar o conteúdo dos registros gerais. A definição de qual dos conjuntos está sendo utilizado pelo programa em execução, é feita por um bit da Palavra de Estado do Processador (PEP).

Embora qualquer dos registros gerais possa ser usado como apontador de pilha ("stack pointer") sob controle do programa, algumas instruções, associadas com tratamento de inter

rupções, usam automaticamente o registro R6 como apontador.Há um registro R6 para cada modo de operação do processador.

O registro R7 é usado pelo processador como Contador de Programa. Cada vez que uma instrução ou índice é lido da memória, o conteúdo de R7 é automaticamente incrementado, passando a apontar a próxima palavras da memória.

Além dos registros gerais existem seis acumuladores de ponto flutuante com 64 bits, ACØ a AC5, referenciáveis por instrução. Nas instruções de ponto flutuante um dos operandos encontra-se em um desses acumuladores e outro geralmente na memória.

2.4 - Modos de Endereçamento

Os dados armazenados na memória são acessados e operados por instruções da máquina que geralmente especificam:

a) A função a ser executada através do código de operação.

b) Um registro geral para ser usado na busca do operando fonte e/ou um registro geral para busca do operando destino.

c) Um modo de endereçamento para ser usado na busca do operando fonte e/ou um modo de endereçamento para busca do operando destino.

Os acessos aos operandos na memória podem ser feitos por byte(8 bits) ou palavra que na UCP tem 16 bits. A palavra de memória consiste de um byte alto e um byte baixo. Os bytes baixos são armazenados nos endereços pares da memória e os altos nos endereços ímpares. Palavras começam sempre em endereços pares. Quando são feitos acessos em palavras consecutivas de memória o registro apontador é sempre incrementado por dois.

Os operandos são lidos da memória com auxílio dos registros gerais de forma direta ou indireta, utilizando um dos oito modos de endereçamento seguintes:

Endereçamento Direto

a) Modo Ø (Registro) - Neste modo qualquer dos registros gerais pode ser usado como simples acumulador e o ope

rando está contido no próprio registro. Como são registros internos ao processador as operações são feitas com maior rapidez.

b) Modo 2 (Auto-incrementado) - Este modo é usado para acessos sequenciais em tabelas de operandos. O conteúdo do registro selecionado é usado para apontar o operando e ao terminar a busca é incrementado por um para instruções que operam com "byte", por dois para instruções de palavra e sempre por dois quando são usados R6 e R7.

c) Modo 4 (Auto - decrementado) - Este modo é usado para acessos na direção oposta a do modo 2. O conteúdo do registro especificado,primeiro é decrementado (por um em instruções de byte e por dois em instruções da palavra ou que usem R6 ou R7), e a seguir usado pera buscar o operando. Estes dois modos de endereçamento, auto-decrementado e auto-incrementado, facilitam operações no modo pilha.

d) Modo 6 (Indexado) - O conteúdo do registro geral especificado é somado ao conteúdo da palavra seguinte a instrução (palavra de índice) para apontar o operando. Neste modo o conteúdo do registro geral pode ser usado como base para calcular uma série de endereços, permitindo acessos randômicos em uma tabela.

Endereçamento Indireto

a) Modo 1 (Registro Indireto) - O conteúdo do registro é usado para apontar o operando.

b) Modo 3 (Auto - incrementado indireto) - Neste modo o conteudo do registro é usado como ponteiro para buscar a palavra na memória que contém o endereço do operando. A seguir o registro é incrementado por dois. Este modo e o seguinte são usados para acessos sequenciais em tabelas constituidas de endereços ao invés de operandos.

c) Modo 5 (Auto - decrementado indireto) - Neste modo o registro é primeiro decrementado por dois e a seguir usado para buscar a palavra que contém o endereço do operando.

d) Modo 7 (Indexado Indireto) - O conteúdo do registro geral é somado a palavra seguinte a instrução (índice) e então o resultado é usado para buscar o endereço do operando. Nem o registro nem o índice são alterados.

2.5- Formato dos Operandos

A Unidade Aritmética opera com dois tipos de números: inteiro e ponto flutuante. Os números inteiros podem ter 16 bits, formato curto, ou 32 bits, formato longo. Exceto nos casos das instruções de multiplicação e divisão, o formato curto é o adotado pelas instruções que operam com números inteiros.Os números são representados em complemento a 2^n. Como uma palavra de memória é composta de 16 bits, um número inteiro ocupa uma ou duas palavras para sua representação, conforme o formato seja curto ou longo respectivamente (fig III-3).

Os números em ponto flutuante tem a forma $M.2^E$, sendo E o expoente e M a mantissa. O expoente é representado em 8 bits podendo assumir valores entre -128 e + 127, entretanto como é sempre armazenado com polarização (soma-se 128 ao valor do expoente),sua representação binária varia entre ø e 255. Este procedimento, embora complique a execução das instruções de multiplicação e divisão flutuantes, simplifica no caso das instruções de soma e subtração que são mais frequentes.

A mantissa é representada em sinal e magnitude e conforme a precisão seja simples ou dupla, possui 24 ou 56 bits respectivamente. Como a mantissa é sempre normalizada, ou seja, $0,5 \leq M < 1$, o bit imediatamente a direita do ponto é sempre um, exceto quando o expoente é zero, aí o número é considerado zero.Este bit mais significativo não é armazenado na memória, sendo inserido durante a execução da instrução. Um número em ponto flutuante é armazenado na memória em duas ou quatro palavras de 16 bits como está ilustrado na figura III-3. A primeira palavra pos

sui 1 bit para o sinal, 8 para o expoente e os bits restantes armazenam a parte mais significativa da mantissa. A parte restante da mantissa é armazenada nas palavras que se seguem.

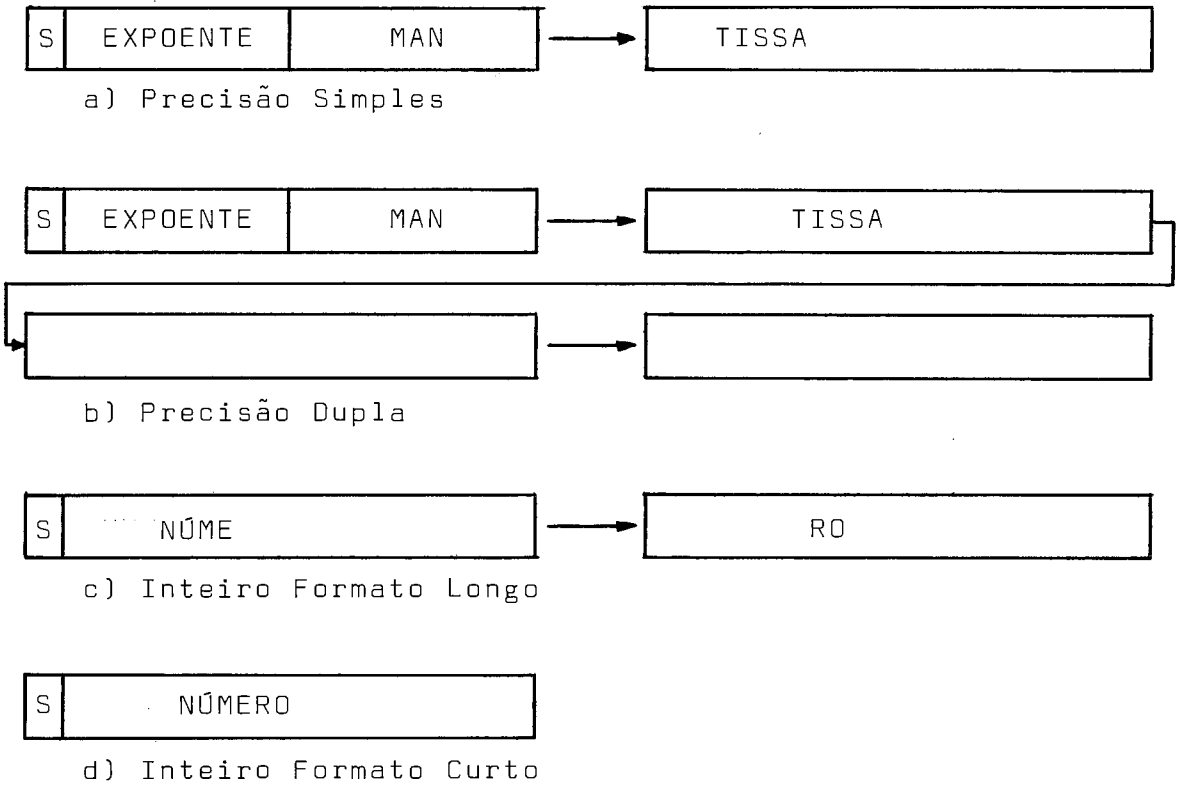


figura III-3 - Formato dos Operandos

2.6- Conjunto de Instruções

A CPU é capaz de executar 86 instruções de diversos tipos. Para facilitar a microprogramação, estas instruções foram divididas em grupos segundo o número e tipo de operandos e a maneira de execução. Os dois grupos básicos são as instruções que operam dados inteiros e as instruções de operandos em ponto flutuante. Com a finalidade de dar uma idéia do conjunto de instruções inteiras e facilitar o entendimento da estrutura dos microprogramas, que será apresentada mais adiante, é mostrada a seguir a classificação adotada e exemplos de algumas instruções. Uma descrição das instruções de ponto flutuante e de seus microprogramas, pode ser encontrada no trabalho de Mário Ferreira Martins[7].

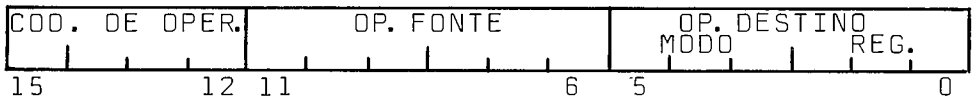
2.6.1- Instruções de Um Operando

Os bits 15 a 6 especificam o código de operação da instrução e os bits 5 a Ø formam um campo de 6 bits, chamado campo de endereçamento destino (fig. III-4a). Ele consiste de duas partes:

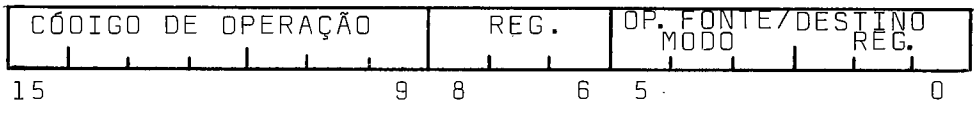
a) Os bits 2 a Ø especificam um dos oito registros de uso geral para ser usado no cálculo do endereço do operando.

b) Os bits 5 a 3 indicam como o registro selecionado será usado para o cálculo do endereço (modo de endereçamento).

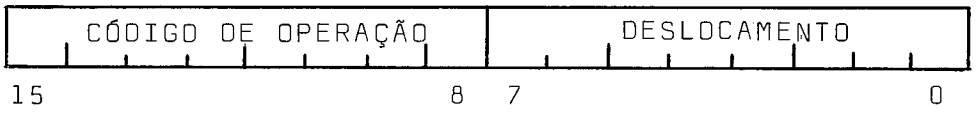
a) Instruções de um operando



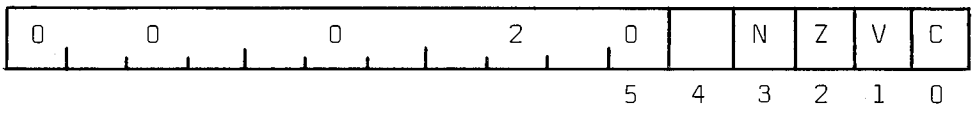
b) Instruções de dois operandos



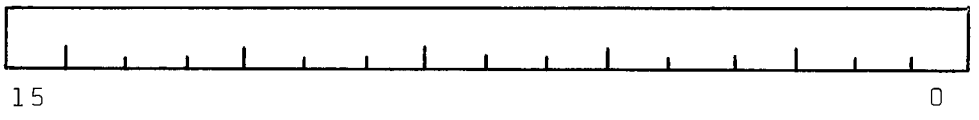
c) Instruções de registro e op. fonte ou destino



d) Instruções de desvio



e) Instruções de posicionamento de código de condição



f) Instruções de execução imediata

fig.III-4 - Formato das instruções inteiras

As instruções de um operando foram divididas, para efeito da microprogramação de sua fase de execução,nos seguintes grupos:

a) Instruções Gerais - São as instruções aritméticas ou de teste do tipo:

CLR - Zera o operando especificado no campo destino.

DEC - Decrementa o operando destino.

TST - Posiciona os códigos de condição segundo o valor do operando destino.

b) Instruções de Deslocamento - Realizam operações de deslocamento aritmético (uma posição ou n posições), ou deslocamentos simples uma posição.

ASR - Desloca o operando destino aritmeticamente uma posição à direita.

ASC - Desloca o operando destino à direita ou à esquerda n posições.

2.6.2- Instruções de Dois Operandos

Operações que implicam no uso de mais de um operando são na sua maioria tratadas por este tipo de instruções. (fig. III-4b). O primeiro operando é chamado operando fonte e o segundo operando é chamado operando destino. O resultado da operação é armazenado na localização apontada pelo campo destino. São instruções que realizam operações aritméticas, lógicas, movimentação e testes, como por exemplo:

MOV - Move o operando fonte para a posição apontada pelo campo destino.

ADD - Soma os operandos fonte e destino e guarda o resultado na posição apontada pelo campo destino.

As instruções de Multiplicação (MUL). Divisão (DIV) e ou-exclusivo (XOR), também são de dois operandos, sendo que um dos operandos está sempre em um dos registros gerais (Fig. III-4.c).

2.6.3- Instruções de Desvio

Estas instruções (fig.III-4d) causam um desvio para uma localização definida pela soma do valor de um deslocamento dado pela instrução multiplicado por 2 mais o conteúdo atual do Contador de Programa. O desvio é feito quando a instrução é incondicional ou é condicional e a condição para o desvio é satisfeita. Os testes para o desvio são realizados sobre o estado dos códigos de condição (Vai um, Transbordo, Número Negativo e Resultado Zero) em instruções do tipo:

BR - Desvio incondicional.

BEQ - Desvia se o bit Z (Zero) é igual a um.

BMI - Desvia se o bit N (Negativo) é igual a um.

2.6.4- Instruções de Execução Imediata

Estas instruções (fig. III-4 e,f) na verdade não formam um grupo em que a fase de execução possui um microprograma comum. Ao contrário, tem cada uma microprogramas independentes, devido ao caráter específico de sua função. São exemplos de instruções deste tipo, as seguinte:

HALT - Para o processador.

WAIT - Dá ao programador uma maneira de liberar a barra de periféricos, enquanto espera uma interrupção.

RESET - Zera todos os registros de comando para periféricos.

2.7- Unidade de Entrada e Saída

2.7.1- Introdução

A função básica da Unidade de Entrada e Saída é realizar a comunicação do processador com os periféricos, sistema de memória e registros internos da UCP. Do ponto de vista do programador, tanto os registros internos da UCP como os registros existentes nos periféricos constituem posições de memória, isto

é, o programador não dispõe de instruções especiais para acessar estes registros, utilizando-se para isso, de instruções idênticas aquelas usadas para acessar à memória.

A comunicação do processador com os registros internos é feita através de vias existentes no interior da própria UCP. Já as operações de E/S referentes a registros de periféricos e posições de memória são realizadas através de duas barras de comunicação externas à UCP.

Uma operação de E/S é iniciada sempre através de uma requisição feita pela Unidade de Controle, que coloca à disposição da Unidade de E/S um endereço em 16 bits, chamado ENDEREÇO VIRTUAL e informações que especificam o tipo de E/S que se deseja fazer (escrita, escrita de byte, leitura de operando, leitura de instrução, etc.). A partir deste instante a operação de E/S é conduzida exclusivamente pela Unidade de E/S, que a executa em quatro etapas.

a) Relocação do endereço virtual de 16 bits gerado pelo processador, para obtenção do endereço real em 18 bits para comunicação com registros de periféricos, ou em 22 bits para comunicação com registros internos ou com o sistema de memória.

b) Decodificação do endereço real gerado, o que permite à Unidade de E/S determinar se a operação iniciada se refere a uma posição de memória, a um registro de periférico ou a um registro interno da UCP.

c) Deteção de erros no processo de formação do endereço real.

d) Realização da operação de leitura ou escrita requisitada.

2.7.2- Sistema de Relocação de Endereços do Processador

Um programa qualquer na memória se encontra dividido em páginas e a cada página está associado um endereço base de relocação. Cada página pode possuir no máximo 128 blocos de

64 bytes e seu endereço inicial na memória pode ser qualquer múltiplo de 64. A cada programa estão associadas oito páginas no espaço de Instruções e oito páginas no espaço de Dados, totalizando 64 k bytes de endereçamento por programa na memória. Para cada modo de operação do processador (Kernel, Supervisor ou Usuário), existem duas tabelas de relocação: uma para o espaço de Instrução outra para o espaço de Dado. Cada tabela é constituida por 8 pares de registros, cada par corresponde a uma determinada página. Este par é constituido por um registro que define o endereço inicial da página e outro que descreve a página, definindo o seu tamanho e os tipos de acessos permitidos. É através deste registro que se consegue fazer proteção de páginas na memória,definindo-as como não residentes ou de leitura exclusiva.

2.7.3- Espaço de Endereçamento

O espaço de endereçamento colocado a disposição do processador compreende 4 M bytes. Os 8 K bytes de endereço mais alto referem-se a registros de periféricos ou registros internos da UCP.Os 264 k bytes seguintes são utilizados tanto pelos periféricos como pelo processador para acessar a memória através do Sistema de Relocação de Endereço da Barra de Periféricos. Os endereços restantes referem-se a posições físicas de memória e são usadas pelo processador para acessar diretamente o Sistema de Memória.

2.7.4- Deteção de Erros

Os erros detetados pelo Sistema de E/S, em função de sua natureza, podem provocar ou não a suspenção imediata da execução da instrução em curso. No primeiro caso, os erros serão causadores de sinalização de ABORTO e no segundo caso sinalização de "TRAP". Ao ser gerada uma sinalização de ABORTO, a execução da instrução é suspensa e inicia-se a execução de uma rotina para tratamento do erro. Quando há sinalização de "TRAP", a instrução é terminada e aí então inicia-se uma rotina de tratamento da ocorrência. Os erros que causam sinalização de aborto

podem ser de três tipos: erros de endereçamento, erros no uso da pilha do processador no modo Kernel e erros de paridade. A sinalização de "TRAP" acontece quando há erro de paridade na palavra seguinte a posição requisitada, ameaça de violação da pilha, falha no sistema de alimentação e erros em operações de ponto flutuante. Também são geradas sinalizações de "TRAP", para permitir ao Sistema Operacional efetuar controle estatístico da frequência e tipos de acessos às páginas residentes na memória, para permitir execução de programas instrução a instrução, facilitando depurações ou para permitir operações de painel.

2.7.5- Interrupções

O sistema possui quatro níveis de interrupção por circuito para realização de operações de E/S com periféricos e sete níveis de interrupção por programa. Uma interrupção é atendida ao final da execução uma instrução, caso não haja ocorrido "TRAP" e caso seu nível de prioridade seja mais alto que o nível de prioridade do processador,definido pela Palavra de Estado do Processador. O nível de prioridade do processador pode variar de Ø a 7. As interrupções por circuito podem estar relacionadas aos níveis 4,5,6 ou 7 de prioridade e as interrupções por programas podem possuir níveis de prioridade de 1 a 7.

Uma descrição detalhada da Unidade de Entrada e Saída, pode ser encontrada no trabalho de tese de Júlio Salek Aude[8].

2.8- Sistema de Memória

O Sistema de Memória em projeto para esta UCP é do tipo "CACHE-BACKING", visando solucionar o problema de uma memória de alta capacidade e alta velocidade. A capacidade do Sistema de Memória é a capacidade da memória "backing",enquanto que a velocidade do sistema se aproxima da velocidade do "cache",que é muito mais rápido que o "backing".

O "backing" utiliza circuitos de memórias dinâmicas da família MOS, sendo o tempo de ciclo da memória em torno

de 500 ns. A capacidade do "backing" pode ser extendida até 4 M bytes.

O "cache" é composto de pastilhas de memória estática da família TTL Schottky. Sua capacidade de armazenamento total é de 8 K bytes e o tempo de acesso se situa em torno de 80 ns. O "cache" é opcional podendo ser desconectado, permenecendo apenas o "backing" como elemento armazenador.

As operações de leitura ou escrita no "cache" só podem ser realizadas pela UCP ou através do Sistema de Relocação de Endereços da Barra de Periféricos (SRBP). As operações de leitura sempre se referem a uma palavra e as de escrita podem ser relativas a palavra ou byte. A escrita no "cache" é feita em dupla palavra quando se transfere o conteúdo de um bloco do "backing" para o "cache". Esta operação é provocada por uma operação de leitura com falha, ou seja quando há requisição de leitura de uma posição de memória que não está no "cache".

No "backing" os ciclos de leitura são sempre realizados em dupla palavra, ou seja o backing libera duas palavras. Estas duas palavras são carregadas no "cache" e caso a requisição tenha partido da UCP ou do Sistema de Relocação de Endereços da Barra de Periféricos somente a palavra requisitada é enviada. Se o pedido veio do controlador de E/S as duas palavras são enviadas.

As operações de escrita podem ser feitas em bytes ou palavras quando requisitadas pela UCP ou pelo SRBP,e em dupla palavra quando requisitadas pelos controladores de E/S.

O circuito de controle do Sistema de Memória além de comandar as operações de leitura ou escrita e os "interfaces" com os três dispositivos que requisitam ciclos de memória, tem ainda as seguintes funções:

a) Estabelecimento de uma escala de prioridades para os atendimentos de pedidos de ciclos.

b) Lógica de determinação da ocorrência de falha ou acerto.

c) Lógica de inicialização do "cache".

d) Deteção e indicação de erros e ocorrências internas

no sistema de memória.

A indicação de erros é feita por meio do acionamento de bits em um conjunto de 6 registros do sistema de memória, que servem de meio de comunicação entre este e o resto da máquina.

Maiores detalhes do Sistema de Memória são apresentados no trabalho de tese de Ageu Cavalcanti Pacheco Junior[9].

IV - ORGANIZAÇÃO DA UNIDADE ARITMÉTICA

1- Introdução

O processador visto da Unidade de Controle é composto de uma série de registros, memórias,caminhos de dados (barras de comunicação), deslocadores e unidades que realizam operações lógicas e aritméticas. A alocação e definição de como serão usados estes recursos de modo a executar a função especificada pela instrução, é o problema do microprogramador ao implementar um determinado conjunto de instruções.

Fazendo-se uma analogia com a programação, o trabalho do microprogramador corresponde a escrever uma rotina, na qual uma função lógica ou aritmética é descrita passo a passo, através de funções elementares (microordens). Na programação, através das instruções, a rotina informa ao processador o que fazer, enquanto que a microprogramação, através das microinstruções, diz como deve ser feito, controlando os recursos da máquina. A microprogramação é análoga a programação convencional, os conceitos de subrotinas, desvios condicionais e incondicionais e laços ("loops"), tem as mesmas conotações em ambos os casos. Entretanto, em microprogramação, aparecem problemas adicionais àquele que pretende preparar microprogramas eficientes.

O microprogramador deve ter conhecimento de todos os recursos que estão a sua disposição, cada registro, caminho de dado, memórias locais e as funções que os deslocadores e unidades lógicas e aritméticas executam. Deve conhecer o grau de paralelismo existente no sistema e obter o máximo rendimento desta facilidade. Se preocupar com o sincronismo entre as diversas tarefas, conhecendo os pulsos gerados pelo relógio central e os circuitos afetados por estes pulsos.É importante também saber a relação entre os tempos de acesso a memória principal e o ciclo de execução das microinstruções, procurando tirar proveito da maior velocidade de execução das microinstruções.Portanto o microprogramador deve ter razoável conhecimento tanto de detalhes do circuito e recursos da máquina, bem como da sua filosofia de implementação. Frequentemente, quando os microprogramas são desenvolvidos em paralelo com o projeto dos circuitos, alterações na organização interna visando maior eficiência são sugeridas pe

los microprogramadores.

Este capítulo tem a finalidade de apresentar os recursos da Unidade Aritmética (UA) que estão disponíveis ao microprogramador. A organização interna da UA é mostrada na fig. IV-1.

2- Unidade Aritmética e Lógica

O principal recurso da UA é a Unidade Aritmética e Lógica (UAL). Capaz de operar dados em 8,16 e 64 bits, a UAL realiza operações de adição, subtração, transferência de dados, complementação e operações lógicas do tipo E, OU, OU-EXCLUSIVO, etc. Sob controle do microprograma, é possível nas instruções inteiras que operam com byte alto do operando, usar somente a parte da UAL que vai dos bits 8 à 15. Isto é conseguido controlando o vai uma que sai do bit sete.

A UAL exige 6 linhas de controle: quatro (S3,S2, S1 e S0) escolhem a função que será executada, uma define se a UAL usará o modo de operação lógico ou aritmético e a última posiciona o vai um. O controle da UAL pode vir de duas fontes: o campo do microprograma CTL UAL que especifica diretamente a operação a ser realizada, ou uma memória de leitura exclusiva (MLE). Esta memória tem por função tornar o controle da UAL dependente do código de operação da instrução. Para isso ela recebe como endereço alguns bits do Registro de Instruções e fornece na saída sinais que são utilizados para comandar a UAL. Desta maneira foi possível reunir em uma mesma microinstrução a fase de execução de diversas instruções. Nesta microinstrução os registros que contém os operando são liberados nas barras A e B e a UAL é controlada pelos sinais que vêm da memória.

3- Memória Rascunho

A Memória Rascunho (MR) possui 16 palavras de 64 bits, sendo dividida em 4 módulos de largura 16 bits (fig.IV-1). A divisão em módulos permite utilizar esta memória tanto para armazenar acumuladores flutuantes de 64 bits como os registros ge

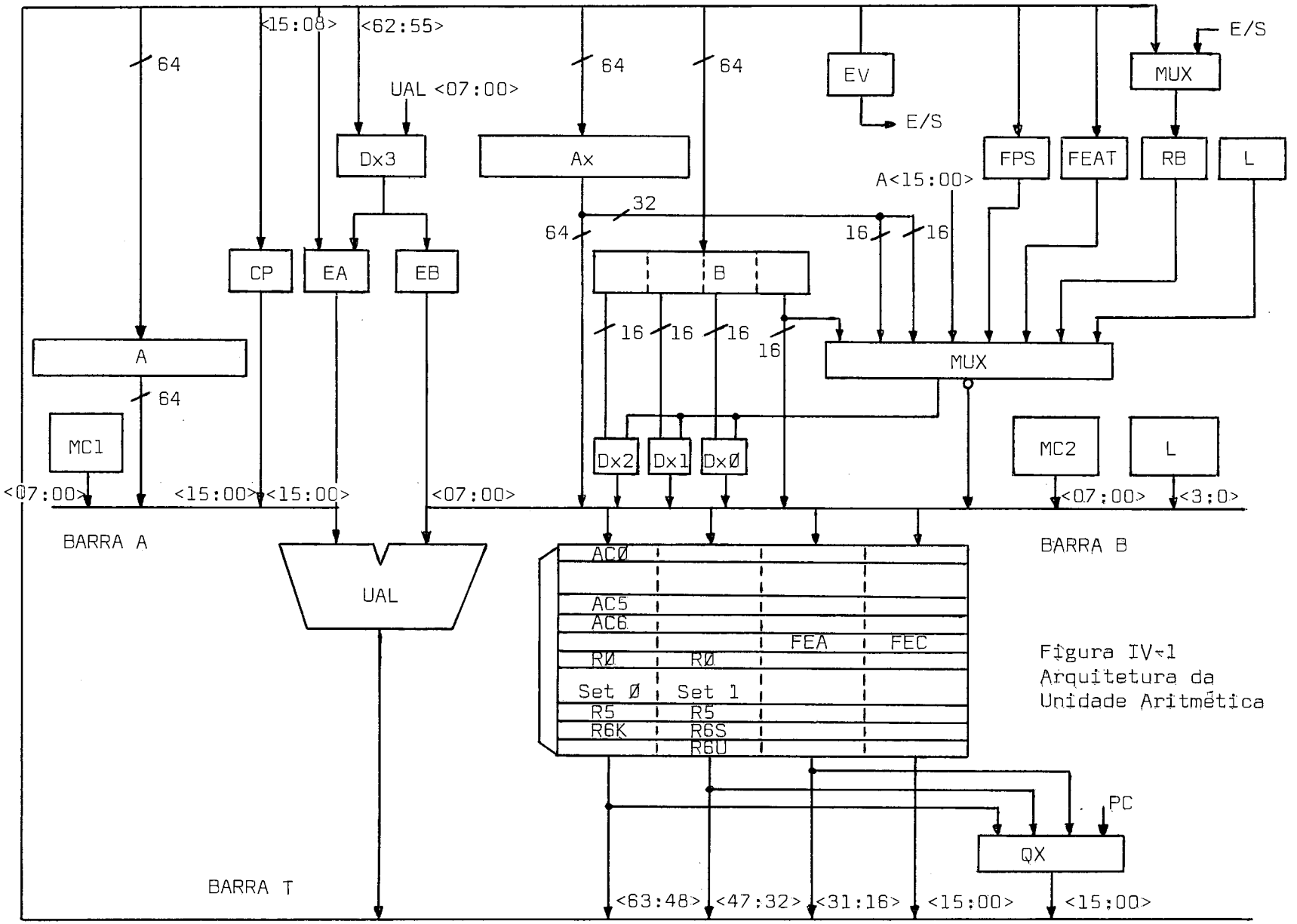


Figura IV-1
Arquitetura da
Unidade Aritmética

rais de 16 bits.

As seis primeiras palavras da MR são utilizadas como acumuladores de ponto flutuante (ACØ a AC5). A sétima palavra é usada como acumulador temporário de 64 bits. Neste acumulador o microprograma carrega da esquerda para direita, cada uma das quatro palavras de 16 bits que compõem o dado de ponto flutuante, quando este é lido da memória. Na oitava palavra são guardados os conteúdos dos registros FEC e FEA, usados durante tratamento de excessões de ponto flutuante. O registro FEC armazena o código da excessão e FEA o endereço da instrução onde ocorreu a excessão.

Na outra metade da memória estão os dois conjuntos de registros gerais e os três apontadores de pilha. O Contador de Programa não é armazenado nesta memória para permitir maior flexibilidade em operações com o seu conteúdo.Da esquerda para direita (figura IV-1), temos no primeiro módulo da memória o conjunto Ø de registros (RØ a R5) e o apontador de pilha (R6) do modo Kernel. No segundo módulo estão o conjunto 1 (RØ a R5) mais os registros R6 dos modos Supervisor e Usuário.

A escrita dos operandos nos acumuladores flutuantes é feita através da barra B <63:00>, enquanto que a leitura de operandos é feita através da barra T <63:00>. Em operações de leitura ou escrita nos registros gerais e execução das instruções de ponto flutuante do tipo "LOAD" e "STORE", que fazem movimentação de operandos entre os acumuladores e memória, os dados passam através de multiplexadores antes de serem lidos ou escritos na MR. Estes multiplexadores (DX2,DX1,DX0,QX, figura - IV-1) permitem que os operandos que estão alinhados na parte baixa das barras (bits <15:00>) possam ser escritos em qualquer dos módulos da MR que estão nas partes altas das barras.

O endereço da MR é composto de 4 bits.O mais significativo que permite escolher entre a metade superior, onde estão os acumuladores flutuantes, e a metade inferior, onde estão os registros gerais, provém diretamente da palavra de controle. Os três seguintes são obtidos através de um multiplexador que, sob controle do microprograma, seleciona uma das quatro fontes abaixo:

a) Bits <Ø2:ØØ> do Registro de Instruções (RI) - São usados para apontar um dos registros gerais. Em instruções inteiras o registro geral lido é usado para cál

culo do endereço do operando destino, e nas instruções flutuantes no cálculo do endereço do operando fonte ou destino.

b) Bits <08:06> do RI - São usados nas instruções inteiras para apontar o registro geral que será usado no cálculo do endereço do operando fonte. Em instruções de ponto flutuante o bit 8 é forçado para zero, e os três bits apontam um dos acumuladores flutuantes.

c) Registro de Instruções <08:07> # 1 (instruções inteiras) ou 0 # RI<07> # 1 (instruções de ponto flutuante) - Esta entrada do multiplexador é usada para apontar o registro fonte V 1,ou acumulador flutuante V 1 (V=função lógica ou) nas instruções que utilizam dois registros, um par outro ímpar, para armazenar os operandos. (Obs.: # símbolo usado para significar concatenação).

d) 1 # END MR <1:0> (Instruções inteiras) ou Ø # END MR <1:0> (Instruções flutuantes) - Entrada do multiplexador usada pelo microprograma para apontar registros gerais ou acumuladores flutuantes em operações de leitura ou escrita. END MR <1:0> é o campo do microprograma usado nestas operações.

4- Registros EA e EB

Os registros EA e EB são usados nas instruções de ponto flutuante para armazenar os expoentes dos operandos e nas instruções inteiras armazenam informações sobre os endereços dos operandos.

O registro EB tem oito bits e é do tipo contador. EA tem 16 bits, mas só os 12 bits menos significativos, usados durante as instruções de ponto flutuante, são do tipo contador. Normalmente, durante a execução destas instruções os expoentes são armazenados nos oito bits menos significativos dos registros. Os bits adicionais do registro EA são usados para detetar

transbordo ou "underflow" do expoente durante as operações de normalização de resultados ou correção de transbordo de mantissa.

Nas instruções inteiras o registro EA é utilizado no cálculo do endereço do operando fonte, permanecendo com o endereço durante toda a fase de execução da instrução. No registro EB é armazenado o bit menos significativo do endereço do operando destino. Com as informações sobre os endereços dos operandos inteiros armazenadas em EA e EB, mais a indicação de que a instrução é de palavra ou não, é possível determinar em qual dos bytes menos significativos da UAL, será realizada a operação requerida pela instrução.

Durante a execução das instruções de ASHC e ASH, o registro EB é utilizado como contador do número de deslocamentos que o operando no registro AX vai sofrer.

Os registros EA e EB ao contrário dos demais,não recebem os dados diretamente da barra T, mas através de multiplexadores (fig. IV-1), isto porque os oito bits menos significativos nos dois registros podem vir da barra T <62:55> ou barra T <07:00>. No início da execução das instruções de ponto flutuante, o expoente a ser carregado em EB ou EA provem da Memória Rascunho bits 62 a 55, via barra T. Durante a fase de execução das instruções, quando são feitas operações com os registros, os dados vem pela barra T <07:00>.

5- Registros A, B e AX

São registros de 64 bits utilizados para armazenar as mantissas dos operandos de ponto flutuante ou como registros de trabalho em instruções inteiras.

Nas instruções inteiras os números são armazenados em complemento a 2 estando alinhados à direita, ou seja, o bit <15> do número coincide com a posição <15> do registro. Nas instruções de ponto flutuante as mantissas começam na posição <59> (bit mais significativo). Se a precisão for simples a mantissa tem 24 bits e estará situada de <59> a <36>, em precisão dupla estará situada de <59> a <04>. Os bits adicionais dos registros são utilizados para deteção de transbordo e operações de Normalização e Arredondamento. Além disso, alguns dos bits adi

cionais se devem ao número mínimo de bits que compõem as pastilhas utilizadas para implementar os registros.

O registro A pode liberar os 64 bits do seu conteúdo na barra A, ou então, os 15 bits menos significativos na barra B, através do multiplexador 8 para 1 (MUX, figura IV-1).Os bits <15:00> do registro A atravessam o multiplexador, para permitir que os resultados de operações aritméticas e lógicas possam ser escritos na memória rascunho, quando necessário.

O registro AX pode liberar todo o seu conteúdo diretamente na barra B, ou somente os 32 bits menos significativos também na barra B, mas através do multiplexador 8 para 1. Este último modo de liberação na barra decorre da necessidade da instrução inteira de deslocamento aritmético ASHC, que é executada em AX. Nesta instrução o operando pode ter 1 ou 2 palavras ocupando AX <15:00> ou AX <31:16> # AX <15:00>. Em qualquer caso, após os deslocamentos o resultado deve ser carregado de volta nos registros gerais. Como já vimos (ver IV-3) a única maneira destes resultados serem carregados na MR é através do multiplexador 8 para 1 e dos duplexadores DX2 e DX1 (figura IV-1), que permitem atingir qualquer módulo na MR.

O registro B é dividido em quatro módulos de 16 bits, sendo que somente o módulo correspondente aos bits menos significativos é liberado diretamente na barra B. Os outros 3 módulos atingem a barra através dos duplexadores DX2, DX1 e DXØ.

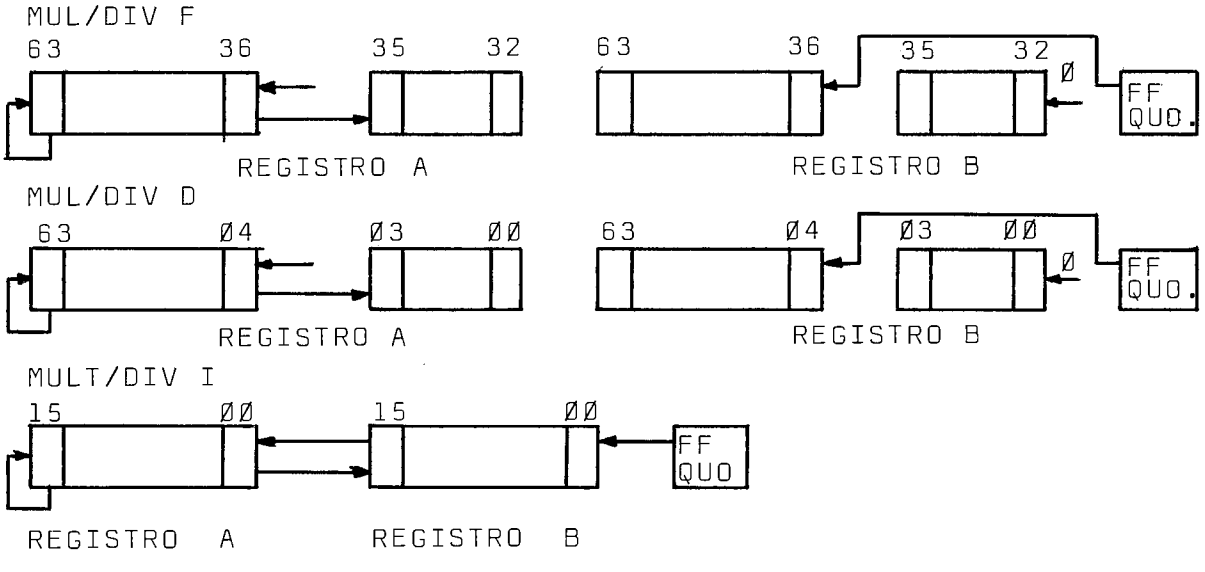


Figura IV-2 - Concatenação dos registros A e B

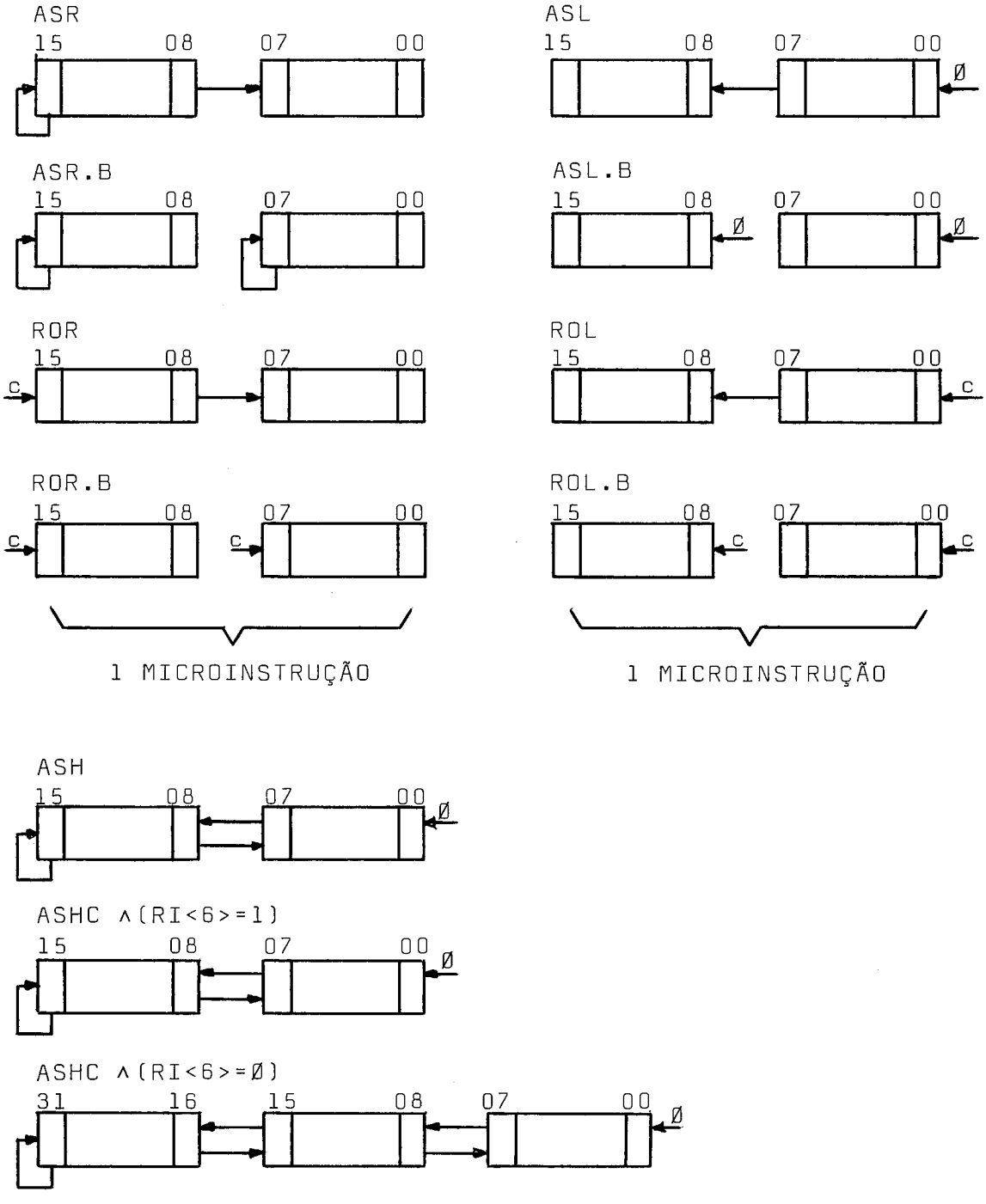


figura IV-3 - Concatenação do Registro AX

Os registros A e B são concatenados para permitir a execução das instruções de Multiplicação e Divisão. A concatenação é feita em função da instrução e dos diferentes formatos existentes (ponto flutuante simples (F) ou duplo (D) e inteiro (I)), conforme está mostrado na figura (IV-2).

Todos os deslocamentos aritméticos ou simples e as execuções das macros NORMALIZE e EQUALIZE são realizadas no registro AX. As possibilidades de concatenação devido ao grande número de instruções que utilizam o registro AX para fazer deslocamentos são muitas. Por esta razão o microprogramador pode escolher entre dois tipos de controle da concatenação: concatenação normal ou condicionada ao tipo da instrução em execução. A concatenação condicional é totalmente comandada por uma MLE. Esta memória recebe bits do RI, suficientes para identificar a instrução em execução e fornece na saída sinais para comandar duplexadores de concatenação colocados entre os diversos módulos em que o registro está dividido. A figura IV-3 mostra as instruções e os correspondentes modos de concatenação do registro AX. Quando da execução de uma dessas instruções, o microprograma necessita apenas liberar a concatenação condicional e dar a ordem de deslocamento à direita ou esquerda conforme a instrução. Isto permitiu que diversas instruções fossem executadas em uma única microinstrução, conforme está indicado na figura IV-3, economizando palavras da memória de controle.

6- Registros EV(Endereço Virtual) e RB(Registro de Barra).

São registros de 16 bits utilizados pela Unidade Aritmética durante comunicação com os registros internos, periféricos ou memória. "RB" serve como um registro "destino",onde são armazenados os dados que chegam a saem da UA. Ele funciona em conjunto com o registro EV, onde é carregado o endereço que será utilizado na operação de E/S. Uma E/S começa quando o chamado endereço virtual de 16 bits gerado pelo processador é carregado em EV. A partir deste instante o endereço sofre um processo de relocação que o transforma no chamado endereço real de 22 bits. Em seguida é feita uma verificação de ocorrência de erros no endereçamento e a decodificação para determinar o destino do endereço:memória, periféricos ou registros internos. Todas estas operações iniciadas a partir da carga em EV são realizadas pela Unidade de E/S independentemente da Unidade de Controle.

7- Registro FPS

Registro de 16 bits que controla o modo de operação da UA em instruções de ponto flutuante, bem como armazena os códigos de condição e dá informações sobre os erros ocorridos durante a execução da instrução anterior. É dividido em campos conforme está mostrado na figura IV-4.

O campo correspondente aos códigos de condição (FPS <03:00>) é posicionado por circuito ao fim de cada instrução, de acordo com o tipo de resultado obtido. O campo seguinte (FPS <07:05> controla os modos de operação, especificando a precisão dos operandos e se deve haver arredondamento ou truncamento nos resultados. O terceiro e o quarto campo, FPS <11:08> e FPS <14> respectivamente, são usados para permitir ou não interrupção do programa no caso de ocorrência das excessões que estão indicadas na tabela IV-1. O último campo (FPS <15>) é um indicador de que ocorreu alguma excessão durante a execução da instrução, entretanto só é posicionado se os bits que liberam interrupção estão ativados. Caso ocorra erro no código de operação ou tentativa de divisão por zero ele é sempre posicionado.

8- Registro FEAT

O registro é utilizado para armazenar o endereço da instrução que está em execução, sendo atualizado cada vez que uma nova instrução inicia sua execução. Os registros FEAT, FEA e FEC possibilitam a recuperação de erros ou excessões ocorridas durante a execução das instruções de ponto flutuante. Quando ocorre alguma excessão ou erro e os bits que liberam as interrupções estão ativados (FIU, FIUV, FIV e FIC), o código do erro (tabela IV-1) é carregado no registro FEC e o endereço da instrução que está armazenado em FEAT e transferida para FEA. Mais tarde no atendimento da interrupção, a rotina que trata o erro pode então descobrir através de FEC e FEA qual foi o erro e em que instrução ocorreu. Deve-se observar que o programador só tem acesso aos registros FEC e FEA.

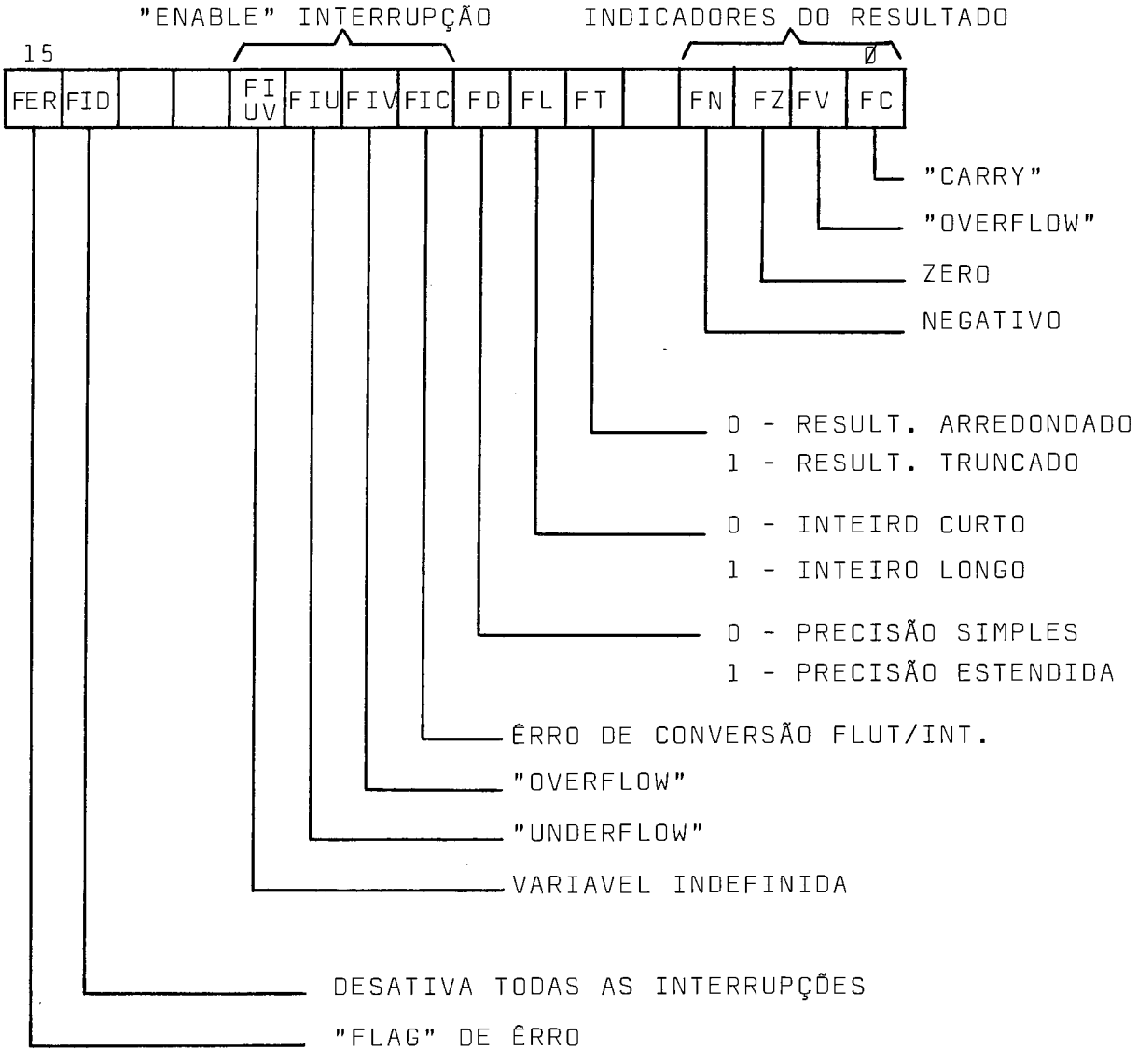


Figura IV-4 - FPS: PALAVRA DE "STATUS" DE PONTO FLUTUANTE

Código	Tipo do êrro
0	não usado
2	código da instrução inválido
4	divisão por zero
6	êrro de conversão flutuante/inteira
8	"overflow"
10	"underflow"
12	variável indefinida

Tabela IV-1 - Códigos das excessões flutuantes

9- Constantes

Na UA há necessidade de várias constantes, que são utilizadas em diversos instantes durante a execução dos microprogramas. Constantes são usadas por exemplo em:cálculos de endereço dos operandos (operações de auto-incremento e decremento); carga de contadores que controlam o número de deslocamentos em registros (instruções de deslocamento aritmético n posições),obtenção dos endereços onde estão os pontos de entrada das rotinas que tratam dos abortos e "traps", etc. Estas constantes são geradas por 3 fontes distintas na UA: MC1, MC2 que são memórias do tipo MLE (memórias de leitura exclusiva) com 32 posições de 8 bits e L, um circuito de lógica combinacional.

A palavra de microprograma possui 6 bits para escolher a constante a ser usada. Dois bits (campo LIB CNST) escolhem uma das três fontes (00-INATIVO, 01-MC1, 10-L, 11-MC2), e os quatro restantes são utilizados no endereçamento de MC2.

Em MC1 estão armazenados os endereços dos vetores utilizados pelas instruções de "TRAP". Estas instruções (EMT, BPT, IOT, TRAP), servem para realizar chamadas a emuladores, monitores de E/S, interpretadores definidos pelos usuários, etc., e funcionam efetivamente como interrupções por programa. Também em MC1 estão os endereços dos vetores de "TRAP" e abortos ocorridos durante o processamento normal. O endereçamento de MC1 é condicional e feito através de um duplexador que seleciona entre bits do Registro de Instruções (RI) e sinais provenientes da UE/S,quem apontará a constante. Os bits do RI identificam qual das quatro instruções de "trap" está em execução e portanto o vetor correspondente, enquanto que os sinais vindos da UE/S identificam que erro ocorreu, e também o vetor. Como o endereçamento de MC1 é condicional, durante a sua utilização o microprograma apenas faz a sua liberação através do campo LIB CNST e espera o resultado da leitura na barra A <7:0>.

MC2 é dividida em duas partes com 16 palavras de oito bits cada uma. Uma das partes é reservada para armazenar constantes necessárias durante a execução das instruções de ponto flutuante MUL e MOD. A outra contém constantes utilizadas pelas instruções restantes. Dos cinco bits que endereçam esta memória, quatro provém diretamente do microprograma e apontam uma

dentre as 16 constantes armazenadas em cada parte da memória. O quinto vem de uma lógica que especifica se a instrução em execução é MUL ou MOD, sendo o sinal que divide a memória em duas partes. Os bits na saída da memória são liberados na barra B<07:00>.

A lógica L é totalmente condicional, e nela o microprograma só atua liberando o seu resultado na barra B<07:00>. Esta lógica gera as constantes 1, 2, 4 e 8 utilizadas durante os cálculos dos endereços dos operandos tanto de ponto flutuante como inteiros. Para isso ela recebe informações do Registro de Instruções sobre os modos de endereçamento, registros gerais, tipo de instrução em execução, etc.

10- Macroinstruções

Mesmo tendo controle microprogramado algumas máquinas incluem também circuitos que realizam tarefas repetitivas (geralmente de natureza aritmética) e que normalmente exigiriam um tempo longo se executadas diretamente por microprograma.

Um exemplo do uso de circuitos extras em substituição a microprogramação e que frequentemente é empregado, ocorre na implementação da multiplicação. A execução desta instrução consiste basicamente, de uma série de testes, somas e deslocamentos. Por meio da microprogramação estas operações seriam realizadas por um laço de microprograma repetido até que todos os dígitos do resultado fossem gerados. Este método tem a desvantagem de ser mais lento que a implementação por circuitos. Isto porque a execução de cada uma das operações básicas na implementação por microprogramação é realizada durante o tempo de execução das microinstruções, enquanto que por circuito o tempo gasto corresponde a duração efetiva de cada operação, que é menor.

Na UA existem quatro circuitos que são usados para reduzir o tempo de execução de diversas operações: circuitos de Normalização, Equalização, Divisão e Multiplicação. As microinstruções que utilizam estes circuitos são chamadas macroinstruções, tendo duração variável conforme a tarefa realizada.Existem dois bits do microprograma que escolhem qual dos quatro circuitos será utilizado.

A macro NORMALIZE foi criada primordialmente pa

ra colocar a mantissa do resultado de uma instrução de SOMA ou SUBTRAÇÃO flutuantes, no intervalo $0,5 \leq M < 1$. Esta tarefa é levada a cabo deslocando o conteúdo de um dos registros de mantissa, no caso AX, para a esquerda e decrementando o mesmo número de vezes o conteúdo do registro de expoente (EA). A macroinstrução termina quando o bit 1 mais significativo da mantissa fica alinhado a direita do ponto.

A macro EQUALIZE também foi criada para utilização nas instruções de SOMA e SUBTRAÇÃO flutuantes. Nestas instruções antes de ser realizada a operação, isto é somar as mantissas, é necessário igualar os expoentes. Isto é feito pela macroinstrução, deslocando para direita a mantissa do menor operando (em AX), e decrementando um contador com a diferença entre os expoentes até que ele chegue a zero.

Verificou-se depois que as macros NORMALIZE e EQUALIZE, poderiam, com pequenas alterações, ser também utilizadas em outras instruções, tais como as de conversão de operandos de ponto flutuante em inteiros e de deslocamento aritmético.

As macros MULTIPLIQUE e DIVIDA são usadas tanto nas instruções inteiras quanto de ponto flutuante, embora os operandos nos dois casos tenham diferentes representações.

11- Registro CP (Contador de Programa)

Registro de 16 bits que faz o sequenciamento dos programas, apontando sempre a próxima instrução a ser executada. O CP também pode ser usado como registro geral (Registro 7), respondendo aos modos de endereçamento como os outros registros. Fisicamente está colocado na barra A<15:00> tendo campos do microprograma para carga e liberação como um registro de trabalho. Durante os acessos à Memória Rascunho em que bits do RI apontam um dos registros gerais para uso em cálculo de endereço de operandos, o registro R7 ou CP, comporta-se como se estivesse logicamente dentro da Memória Rascunho, sendo transparente ao microprogramador a sua real posição.

V - ORGANIZAÇÃO DA UNIDADE DE CONTROLE

1-Introdução

O funcionamento da Unidade de Controle da UCP é muito parecido com o modelo proposto por Wilkes. Na UC a cada ciclo de máquina uma palavra é lida da Memória de Controle e armazenada no Registro de Microinstrução para ser usada durante todo o ciclo. Cada palavra corresponde a uma microinstrução, sendo dividida em campos e cada campo controla pontos específicos da máquina. Campos do microprograma que são usados durante o início da execução da microinstrução e antes que um novo endereço seja carregado no Registro de Endereços da Memória de Controle não são armazenados.

Como base para cálculo do endereço da próxima microinstrução a ser executada é usado um campo do microprograma de 9 bits, chamado de Endereço Base. Este endereço pode ser alterado sob controle do microprograma, permitindo a execução de dois tipos de desvio. Quando é necessário testar o estado de um determinado ponto do processador, por exemplo se o resultado da última operação foi igual a zero, é utilizada uma lógica especial de testes. Se a escolha da próxima micro-rotina depende da instrução que está em execução, o desvio é feito testando-se sinais da saída da lógica que decodifica as instruções. O novo endereço após ser montado na lógica de formação de endereços é carregado no Registro de Endereços da Memória de Controle. Esta carga é feita durante a execução da microinstrução, e portanto a busca da próxima palavra é feita em paralelo com a execução da atual.

2- Memória de Controle

Considerável esforço foi feito para diminuir tanto o número de palavras necessárias ao controle como a largura da palavra do microprograma, procurando deste modo reduzir o custo da Memória de Controle. A maioria das tentativas para diminuir a largura da palavra se concentrou em achar a melhor maneira de reunir as diversas microordens isoladas em campos de controle. Cuidado, teve de ser tomado, em não agrupar em um único campo, microordens que poderiam ser usadas na mesma microinstrução, ou em

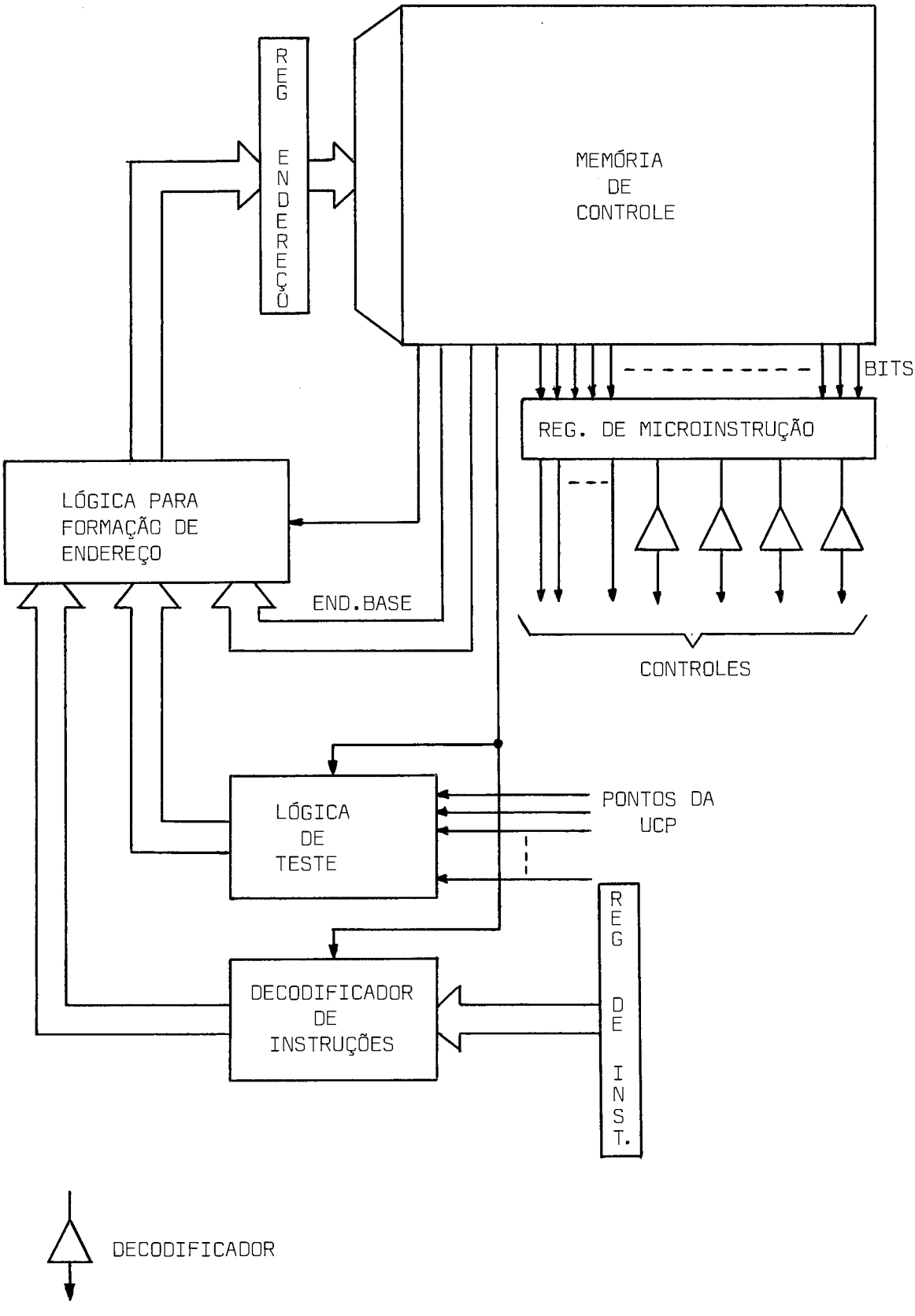


Fig V-1 - Arquitetura da Unidade de Controle

complicar desnecessariamente a decodificação destes campos, com isso degradando a velocidade da máquina.

Para diminuir o número de bits, também foi usada a técnica de permitir que um campo tivesse dupla função.Por exemplo, a UCP executa na mesma Unidade Aritmética, operações sobre dados em ponto flutuante e inteiros. Como consequência, haveria campos que somente seriam utilizados em operações com um desses dois grupos de dados. Para estes campos foi definida uma dupla função, isto é, controlam pontos diferentes da máquina conforme a instrução seja inteira ou flutuante. Um exemplo é o campo que controla as duas Palavras de Estado do Processador. Há na UCP duas Palavras de Estado, uma para instruções de ponto flutuante e outra para instruções inteiras. O controle destas palavras (carga de novos valores ou alterações em bits) é feito através de um único campo, que atua em uma ou outra conforme o tipo da instrução. Neste caso foi adicionada alguma complexidade na decodificação das microordens, na busca de reduzir o tamanho da palavra.

Na tentativa de diminuir o número de palavras da memória de controle, os microprogramas foram estruturados de maneira a aumentar as micro-rotinas que pudessem ser compartilhadas por várias instruções. Como resultado foi necessário um aumento nos circuitos de desvio, para permitir que ao fim da execução de cada micro-rotina, o microprograma possa determinar, em função da instrução atual, a próxima a ser executada.

Um outro artifício usado para reduzir o número de palavras, foi agrupar em uma única micro-rotina, a fase do microprograma, onde é executada a operação lógica ou aritmética especificada no código de operação de várias instruções. Para realizar estas operações a Unidade Aritmética usa uma lógica combinacional complexa (UAL, Cap.IV-2), capaz de realizar um grande número de funções lógicas e aritméticas (SOMA, SUBTRAÇÃO, OU, E, etc.). Esta lógica é normalmente controlada por um campo do microprograma de seis bits, que especifica qual a função a ser executada durante a microisntrução. Fazendo-se que o controle da UAL pudesse também ser dependente do código de operação da instrução, foi possível agrupar a execução de diversas instruções em uma micro-rotina.

O uso de flip-flops de estado também foi uma téc

nica bastante empregada. Ao microprograma é permitido desviar sobre uma condição interna da máquina, mesmo depois que ela tenha ocorrido, e para isso a condição é armazenada em flip-flops de estado. Deste modo, frequentemente, é possível fazer um desvio diversas microinstruções após a ocorrência da condição, evitando-se duplicação de palavras de controle.

Depois deste trabalho de minimização a memória de controle ficou com 512 palavras e largura igual a 113 bits. Nesta memória são armazenados microprogramas para controlar a Unidade Aritmética e Unidade de Entrada e Saída. A memória será implementada com pastilhas de memória de leitura exclusiva de 512 palavras de 4 bits e tempo de acesso típico igual a 40 ns.

3- Palavra de Controle

A palavra de controle é composta de microordens, que podem ser codificadas formando um campo de controle ou não. Para facilitar, os bits isolados da palavra de controle usados como microordens simples, também serão chamados de campos de controle. Com o propósito de aumentar a flexibilidade e consequentemente dar ao programador microinstruções mais poderosas e consistentes com os recursos da máquina, a palavras do microprograma possui campos de controle separados para cada recurso do processador (UAL, Registros, Memória Rascunho, etc.). Isto possibilita que a maioria das operações simultâneas possam ser especificadas, embora nem todas tenham sentido.

Alguns dos campos de controle podem ser logicamente grupados segundo a função que executam dentro do processador. Dentro deste critério podemos criar os seguintes grupos:

a) Campos de Liberação - Controlam a transferência de dados e endereços entre registros, unidade aritmética e lógica e outras unidades, através de barras de comunicação internas ao processador.

b) Campos de Seleção - Estes campos fazem o controle dos diversos multiplexadores existentes no processador, selecionando qual registro vai atravessá-lo e ser liberado na barra de comunicação associada ao

multiplexador.

c) Campos de Controle - Definem uma grande variedade de funções que incluem a operação da UAL, funções de contagem e carga em registros, seleção da macroinstrução que vai ser executada, etc.

d) Campos para Comunicação e Sincronismo com a Unidade de Entrada e Saída - No processador as operações de E/S são realizadas em paralelo com a execução dos microprogramas, com o objetivo de diminuir o tempo de execução das instruções. Quando é necessário uma E/S, a Unidade de Controle através destes campos, define o tipo de operação que deve ser feita (escrita ou leitura) e os tipos de operandos envolvidos (dados, endereços ou índices) e continua a execução da microprograma até que seja obrigatório parar. Em operações de leitura o avanço é feito até que a informação requisitada seja necessária ao prosseguimento do microprograma. Em escritas o microprograma avança até uma nova operação de E/S, que só pode ser realizada após uma verificação com a Unidade de Entrada e Saída se ocorreu durante a operação anterior algum tipo de erro. No caso afirmativo o microprograma é desviado para microrotina de tratamento do erro, caso contrário o microprograma é liberado pela Unidade de Entrada e Saída, para continuar seu curso normal.

e) Campos de Endereçamento - Os bits destes campos fazem o sequenciamento das microinstruções. Cada palavra do microprograma possui um campo de 9 bits, chamado Endereço Base, que é usado para apontar a próxima microinstrução a ser executada. O número de bits no endereço base é suficiente para apontar qualquer palavra da Memória de Controle. Associado ao endereço base, existe um campo que seleciona pontos da máquina para serem testados e conforme o resultado do teste o endereço base é alterado, modificando a sequência das microinstruções. Além desse tipo de desvio o

microprogramas também controla o circuito decodificador de instruções, que gera sinais a serem usados na formação de endereços (Ver lógica de Formação de Endereços, Cap. V-4.2).

A tabela V-1 apresenta os campos do microprograma, sua função, número de bits e a Unidade a que se destina.

MNEMÔNICO	DESCRIÇÃO	DESTINO	Nº DE BITS
LIB A	BARRA A ← A <63:00>	UA	1
LIB B	BARRA B ← B <63:00>	UA	1
LIB AX	BARRA B ← AX <63:00>	UA	1
LIB EA	BARRA A <15:00> ← EA <15:00>	UA	1
LIB EB	BARRA B <07:00> ← EB <07:00>	UA	1
LIB CNST	CONTROLA LÓGICAS GERADORAS DE CONSTANTES	UA	2
LIB PC	BARRA A <15:00> ← PC <15:00>	UA	1
LIB BR	BARRA A <15:00> ← BR <15:00>	UA	1
LIB MUX	0 = SEL MUX; 1= SEL MISC	UA	1
LIB MAC	Ø = ZERA AX; 1= CTL MAC	UA	1
LIB DECOO	CONTROLA LÓGICA DE DECOOIFICAÇÃO	UC	2
LIB CHAVES	LIBERA CHAVES DO PAINEL	UE/S	1
END MR	ENDEREÇO DA MEMÓRIA RASCUNHO	UA	3
END BASE	ENDEREÇO DA PRÓXIMA PALAVRA	UC	9
SEL MISC	LIBERA CONSTANTES, ETC	UA	3
SEL MUX	LIBERA DIVERSOS REGISTROS NA BARRA B	UA	3
SEL END MR	SELEÇÃO DE ENDEREÇOS DA MEM RASCUNHO	UA	2
SEL QX MR	SELEÇÃO DO QUADRUPLEX (QX)	UA	2
SEL MOO MR	SELECIONA MÓDULO DA MEM RASCUNHO	UA	4
SEL TROC BYTE	TROCA DE BYTES DO REGISTRO B	UA	1
SEL CTL UAL	SELECIONA QUEM CONTROLA UAL	UA	1

MNEMÔNICO	DESCRIÇÃO	DESTINO	Nº DE BITS
SEL CTL MR	CONTROLE DA MEMÓRIA RASCUNHO	UA	1
SEL CONC	CONCATENAÇÃO DOS REGISTROS A,B,AX	UA	2
SEL COND DESV	SELECIONA PONTOS P/ TESTE	UA/UC	6
CTL A	CONTROLE DO REG. A	UA	2
CTL B	CONTROLE DO REG. B	UA	2
CTL AX	CONTROLE DO REG. AX	UA	2
CTL EA	CONTROLE DO REG. EA	UA	2
CTL MAC	ESCOLHE TIPO DE MACRO	UA	2
CTL UAL	FUNÇÃO DA UAL	UA	6
CTL PS	CONTROLE DAS PEP	E/S	3
P COD FZ/FN	POSICIDNA CÓDIGOS FZ E FN	UA	3
P COD COND	CARGA DDS CÓDIGOS OE CONDIÇÃO INTEIROS	UA	3
DURAÇÃO	DEFINE Nº DE TEMPOS DA MICRO	UC	1
PARE	PARA O MICROPROGRAMA	UC	1
FIM INST.	INDICA FIM DA INSTRUÇÃO	E/S	1
ACK	ZERA FLIP-FLDPS DE ERRO	E/S	1
DEF INST	DEFINE INSTRUÇÃO EM EXECUÇÃO	E/S	2
MISC E/S	CONTROLES PARA UE/S	E/S	3
DEF E/S	DEFINE TIPO DE E/S	E/S	3
VAL E/S	VALIDA E/S INICIADA	E/S	1
RAP E/S	INDICA QUE END. JÁ FOI RELOCADO	E/S	1
INT E/S	INTERROMPE E/S INICIADA	E/S	1
ESP I/D	DEFINE D ESPAÇO	E/S	1
VRF LIM STACK	UE/S DEVE VER LIMITE DO STACK	E/S	1
E/S FINAL	INDICA ÚLTIMA E/S	E/S	1
CTL FF SINAL	POSICIONA FFS DE SINAL FFSA E FFSB	UA	3
CTL FF ZUAL	POSICIONA FF OE ZERO	UA	2

MNEMÔNICO	DESCRIÇÃO	DESTINO	Nº DE BITS
WAIT	PARA UCP ESPERANDO VETOR INTERRUPÇÃO	UC	1
CAR EB	CARGA DE EB	UA	1
CAR CP	CARGA DE CP	UA	1
CAR RI	CARGA DO REG. INSTRUÇÕES	UA	1
CAR FEAT	CARGA OO FEAT	UA	1
CAR MR	CARGA NA MEM. RASCUNHO	UA	1
CAR RB	CARGA DE RB	UE/S	1
CAR EV	CARGA DE EV	UE/S	1
CAR BRQ	CARGA DE INDICADORES DE ERRO	UE/S	1
ZERA A	ZERA REG. A	UA	1
ZERA B	ZERA REG. B	UA	3
ZERA AX	ZERA REG. AX	UA	3
ZERA EB	ZERA REG. EB	UA	1

As classificações mais comuns para as microinstruções, as descrevem como verticais e horizontais quanto ao seu formato. No entanto, o comprimento da palavra por si só não é suficiente para determinar se a máquina tem microprogramação horizontal ou vertical, desde que, fatores mencionados anteriormente, tais como, codificação de campos e dupla função para campos, afetam diretamente o comprimento da palavra. Contudo, comprimentos de microinstruções longas e curtas tem sido usados para caracterizar máquinas, frequentemente associados a microprogramação horizontal e vertical. Uma boa distinção entre as duas classificações está na capacidade relativa de cada uma exercer controle detalhado sobre os circuitos das máquinas. Esta capacidade é conseguida em alto grau na microprogramação horizontal, e uma consequência normal é uma palavra larga com muitos campos de controle, necessários para comandar todos os recursos. Aplicando estas definições à palavra de microprograma do processador podemos clas

sificá-la como tipicamente horizontal, visto que seus campos permitem um controle detalhado de todos os recursos, possibilitando alto grau de paralelismo.

4- Sequenciamento da Memória de Controle

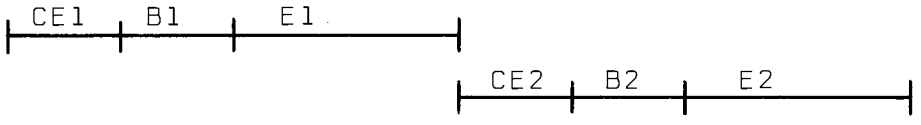
O sequenciamento da Memória de Controle é feito de maneira similar ao modelo de Wilkes. Conforme a fig. V-1 mostra, um dos campos é usado para servir de base para o cálculo do novo endereço, podendo ser modificado por condições da máquina, alterando o fluxo normal da microprogramação. Há dois aspectos diferentes no sequenciamento a serem discutidos, o primeiro diz respeito a sincronização da busca da microinstrução e o segundo está ligado a formação do endereço.

4.1- Sincronização da Busca da Microinstrução

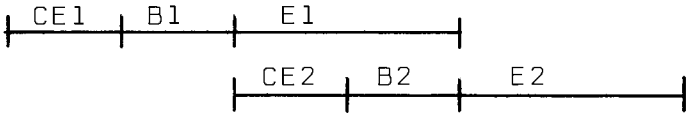
Há dois tipos de técnicas usadas para implementar a busca da microinstrução: busca serial e busca paralela. No primeiro caso a busca só é iniciada quando a execução da microinstrução corrente termina. Deste modo, todas as informações necessárias para determinar qual a próxima microinstrução já estão disponíveis (fig. V-2a). No sistema paralelo a busca é feita concorrentemente a execução da atual, com vantagens óbvias no desempenho. Desvios condicionais entretanto, causam problemas, desde que informações necessárias ao desvio, por exemplo, resultado de uma operação na UAL igual a zero, estão disponíveis somente no final da execução da microinstrução. São utilizadas frequentemente dois tipos de soluções; a primeira consiste em fazer uma previsão do endereço mais provável da próxima microinstrução e iniciar a sua busca. Se a previsão for correta não foi perdido nenhum tempo, caso contrário, um ciclo foi perdido e é iniciada a busca de microinstrução certa,(fig. V-2c).Outra solução consiste em armazenar a condição a ser testada,caso ela seja volátil, executar uma nova microinstrução, sem fazer nenhum desvio, e ao final desta microinstrução desviar. Esta solução requer circuito para armazenar as condições e trabalho para estruturar os micro

programas de modo a fazer com que esta nova microinstrução, em que o desvio é feito, realize alguma tarefa útil. Normalmente, uma simples troca de ordem entre a micro que faz a operação a ser testada (onde deveria ser feito o desvio) e a micro anterior resolve o problema.

a) Busca em série



b) Busca em paralelo



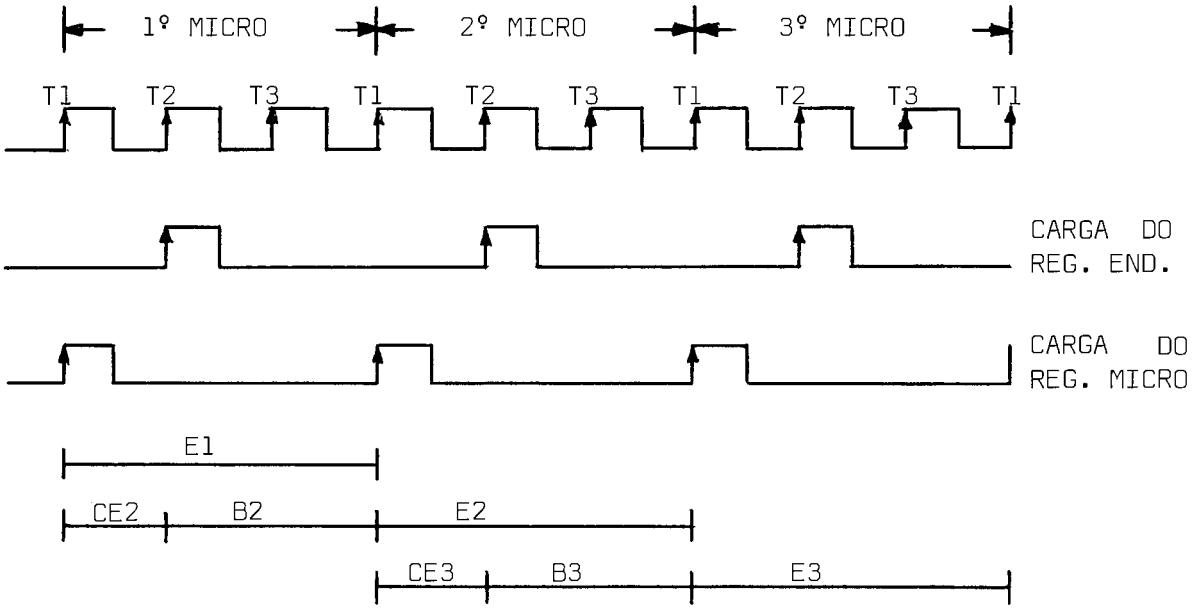
c) Busca em paralelo/Desvio feito com previsão e erro

CE - Cálculo do Endereço
B - Busca
E - Execução

Fig. V-2 - Sincronização de Busca das Microinstruções

Na Unidade de Controle o acesso da próxima micro instrução é feito em paralelo com a execução da microinstrução atual. Caso seja necessário um desvio, a condição é armazenada e na micro seguinte testada. Desnecessário dizer que isso pode trazer problemas ao microprogramador, que gostaria de desviar logo que a condição fosse disponível. Todavia a solução alternativa, como vimos, aumenta o ciclo da máquina. Na microprogramação das instruções inteiras e de ponto flutuante da UCP, os problemas que surgiram foram contornados em quase todos os casos, provando

ser esta técnica uma boa solução para a UCP.



E - Execução

CE - Cálculo do Endereço

B - Busca

Fig. V-3 - Sincronização da busca de microinstrução na UCP

A fig. V-3 mostra o diagrama de tempos para busca e execução de microinstruções na Unidade de Controle. A primeira linha mostra as marcas de tempo geradas pelo relógio central da máquina. Cada período da onda tem 40 ns e a execução de uma microinstrução dura três períodos de 40 ns, chamados T1,T2 e T3 (ver Circuito de Sincronização - Cap. V-6). Em T1 a palavra de controle lida da memória é carregada no registro de micropro grama e imediatamente tem início sua execução.Nesta mesma marca de tempo começa o cálculo do endereço da próxima microinstrução a ser executada. O resultado deste cálculo é carregado no registro de endereço em T2, iniciando uma leitura na memória de controle. A execução da microinstrução continua, passa por T3 e em T1 é carregado uma nova microinstrução, reiniciando-se o ciclo.

4.2- Formação do Endereço

Um grande número de técnicas podem ser usadas no processo formação do endereço da próxima microinstrução a ser executada. Todavia elas podem ser grupadas em um número pequeno de técnicas mais gerais com modificações específicas para cada implementação.

A técnica original de Wilkes de incluir o endereço da próxima microinstrução em um campo da palavra de controle tem sido usada frequentemente, com diferenças na maneira de implementar os desvios condicionais. Uma outra técnica básica usada é fazer com que o registro de microinstrução possa ser incrementado, e os microprogramas sejam executados em endereços colocados sequencialmente na memória de controle. Neste esquema é necessário incluir a possibilidade de desvios incondicionais, o que não é necessário no modelo de Wilkes já que cada microinstrução realiza efetivamente um desvio incondicional.

Quando a microinstrução especifica o próximo endereço, o normal é ser incluido na palavra de controle um ou mais campos que indicam testes a serem feitos. Estes testes modificam bits do endereço especificado, alterando a sequência do microprograma. Portanto, efetivamente é realizada uma instrução de desvio sobre condição, além dos comandos normais especificados nos outros campos da palavra. Outra técnica é a microinstrução incluir na palavra mais de um endereço como alternativa, e campos de testes serem usados para fazer a seleção do próximo endereço.

As máquinas com microprogramação horizontal, que podem interpretar as instruções com poucas microinstruções, usam mais frequentemente o modelo de Wilkes de dar o endereço da próxima instrução devido a alta incidência de desvios condicionais. As máquinas verticais que tem as microinstruções parecidas com instruções de máquina e microprogramas armazenados de maneira bastante sequencial, usam o esquema de incrementar o registro de microprograma.

4.2.1- Lógica de Desvios

Na Unidade de Controle foi usado basicamente o

modelo de Wilkes para fazer o sequenciamento dos microprogramas. Um campo (END BASE) de 9 bits é usado para apontar a próxima microinstrução, possibilitando alcançar qualquer palavra da memória de controle. Este campo corresponde a matriz B no modelo de Wilkes. As diferenças entre os dois modelos estão na maneira de fazer os desvios condicionais. No modelo de Wilkes, o resultado do teste seleciona entre dois endereços qual será o próximo a ser executado. Na UCP, foi usada a técnica mais comum de incluir campos na palavra de controle que selecionam pontos da máquina para serem testados. Conforme o resultado do teste, alguns bits do Endereço Base são modificados alterando a sequência dos microprogramas. No processador estes testes podem modificar os bits 5 e 4 do endereço base. São permitidos desvios binários, caso em que apenas um dos bits é modificado pelo teste, e desvios quaternários em que duas condições independentes são testadas e os dois bits são modificados. Portanto, quando é feito um desvio,os endereços selecionados podem diferir de $(20)_8$, $(40)_8$ e $(60)_8$. Associado a cada um destes dois bits, existe uma lógica que permite ao microprogramador escolher, através do campo SEL COND DESV (6 bits), um entre 64 pontos diferentes da máquina para serem testados (fig. V-4).

Para exemplificar o funcionamento do circuito de desvio tomemos uma microinstrução que necessite desviar sobre o sinal do resultado de uma operação realizada na UAL, e suponhamos que esta condição está incluida no conjunto que pode modificar o bit 5. Para que o teste possa ser realizado, é necessário que o campo SEL COND DESV selecione a condição e que o bit 5 do endereço base esteja inativo (em zero). O bit do endereço inativo permite que ele seja alterado. Como o campo SEL COND DESV é usado para selecionar condições para os dois bits ao mesmo tempo, o bit 4 deve estar ativo (em um) inibindo qualquer sinal que venha da lógica de seleção deste bit. Em desvios quaternários, os bits 4 e 5 do endereço base devem estar inativos, permitindo que as condições em teste escolham uma entre quatro posições da memória de controle para ser executada.

Na verdade, podemos resumir o funcionamemto da lógica de desvios, dizendo que a condição selecionada, o sinal do resultado no exemplo, é usada, a menos de uma inversão, como bit 5 do endereço da próxima palavra do microprograma.

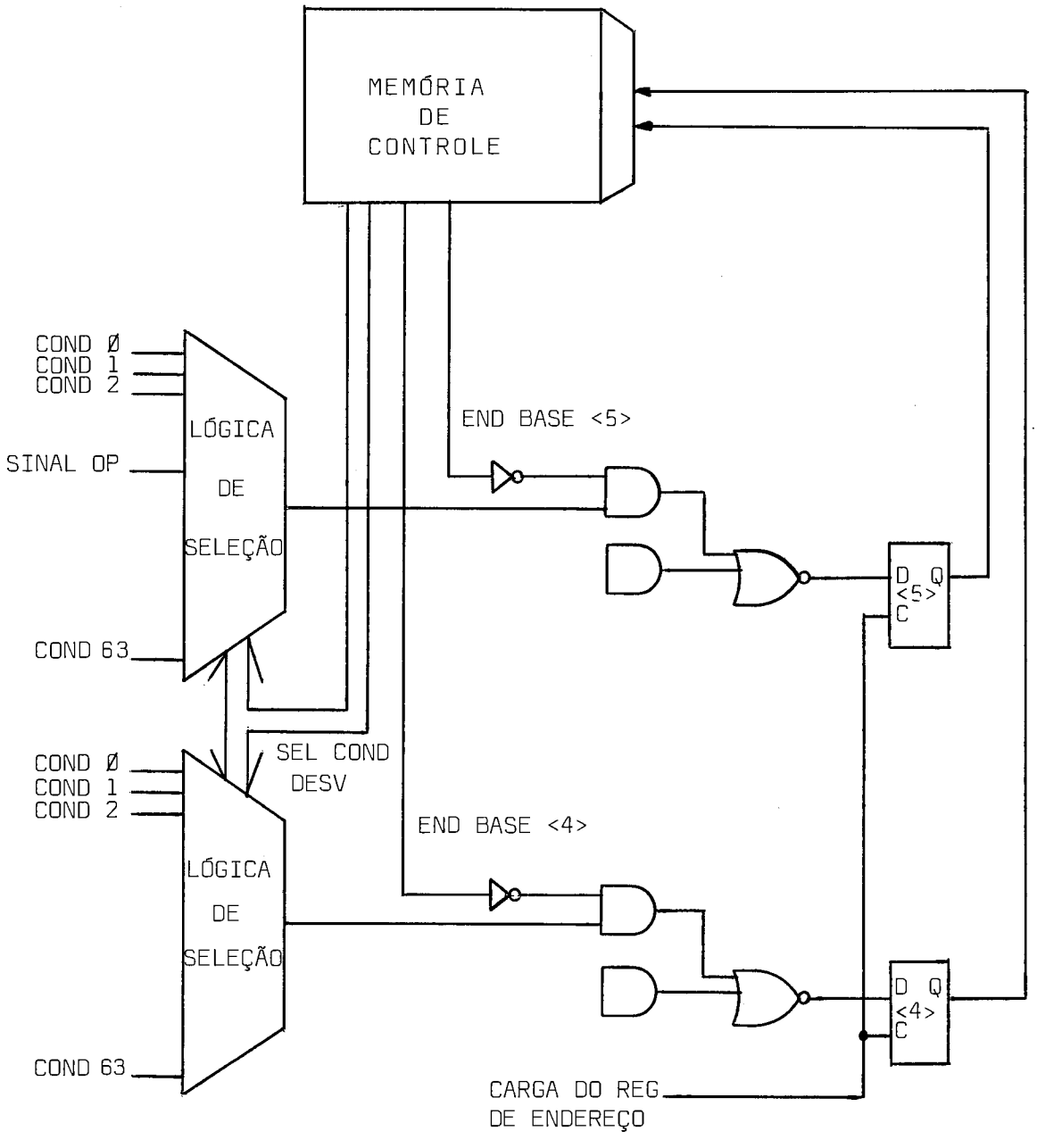


Fig. V-4 - Lógica de Desvios Binários e Quaternários

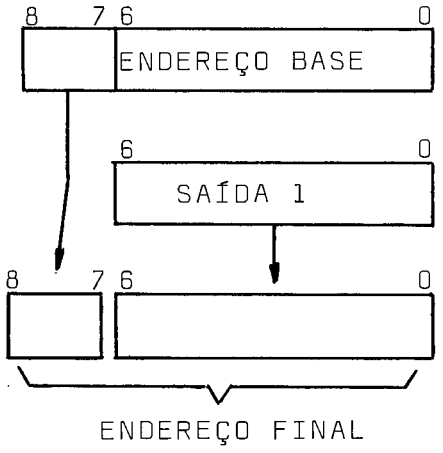
A técnica de desvios binários e quaternários é em alguns casos restritiva, especialmente quando é necessário decidir entre mais de quatro micro-rotinas,qual deve ser a próxima a ser executada. Daí a necessidade de haver desvios de ordem mais alta. Em operações de painel, por exemplo, é feito um desvio para determinar entre nove micro-rotinas qual executa a função pedida pelo operador. No esquema de desvios utilizado neste caso, cinco bits do endereço base (7,6,2,1,Ø) são substituidos

por um código binário que representa a função pedida pelo operador, formando então o endereço inicial da micro-rotina que executará a função. A maneira normal seria executar sucessivamente microinstruções de teste até determinar a função, e aí iniciar a sua execução. Para o painel, a aplicação de desvios de ordem mais alta serviu, principalmente, para economizar microinstruções, já que as operações são por natureza lentas. Um exemplo do uso de desvios de ordem mais alta para acelerar a execução dos microprogramas é dado a seguir com a descrição do decodificador de instruções.

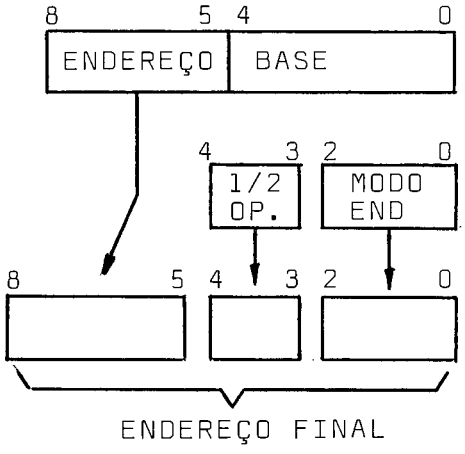
4.2.2- Decodificador de Instruções

O circuito decodificador de instruções é uma lógica puramente combinacional, constituida de um conjunto de memórias de leitura exclusiva programáveis e circuitos lógicos auxiliares, tendo como função decodificar os diversos campos do código de operação da instrução e detetar possíveis erros na especificação destes campos. No circuito existem duas saídas diferentes; a primeira fornece um código binário de 7 bits, que identifica a instrução que está em execução (saída 1); a outra saída dá em código binário de 5 bits informações que especificam se a instrução é de um ou dois operandos e os modos de endereçamento usados para buscá-los (saída 2). No instante em que é feito um desvio que utilize o circuito de decodificação, uma de suas duas saídas, escolhida conforme veremos a seguir, substitui os bits correspondentes do endereço base, formando o endereço da próxima microinstrução a ser executada. A fig. V-5 mostra como é a composição do endereço.

Estas saídas são usadas em desvios, participando da formação de um novo endereço, em dois instantes diferentes durante a execução dos microprogramas. A primeira vez ocorre logo que termina a fase de busca da instrução e tem início a sua fase de execução. Neste ponto, a saída que é usada para formar o novo endereço depende do tipo da instrução que está em execução. Em instruções de execução imediata, isto é, aquelas que não necessitam de mais nenhum acesso a memória para busca de operandos, é usada a saída 1 (fig.V-a). Isto porque o novo endereço é função



a) Desvio sobre código de operação



b) Desvio sobre tipo da instrução e modos de endereçamento

Fig.V-5 - Composição do Endereço Base e saídas do decodificador de instruções.

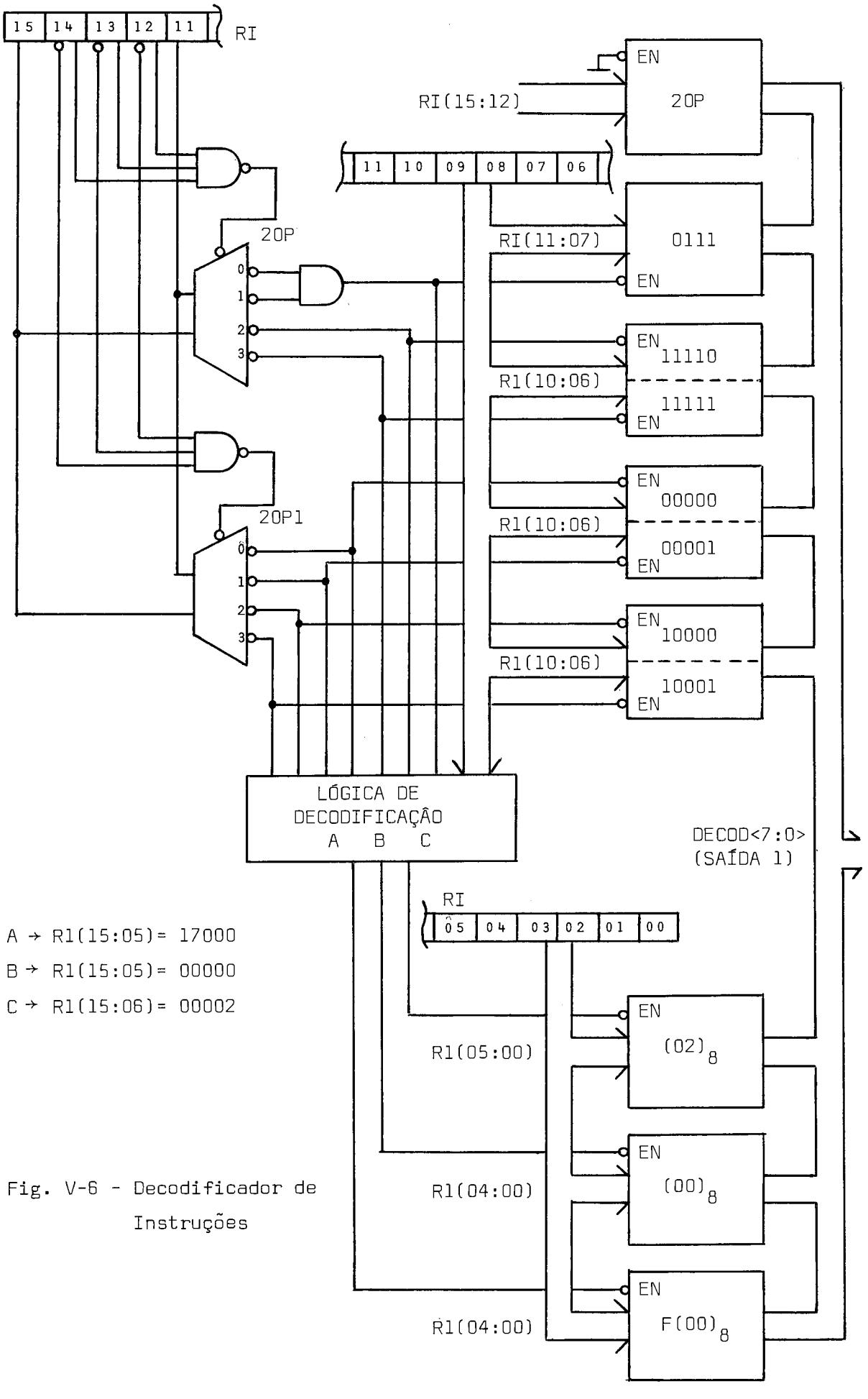
exclusiva do código de operação da instrução, devendo apontar a sua micro-rotina de execução. Em instruções de um ou dois operandos o novo endereço é função tanto do código de operação como dos modos de endereçamento, já que antes da fase de execução propriamente dita, há ainda a busca de operandos. Neste caso é usada a saída 2 do circuito de decodificação (fig.V-5b).

A lógica de decodificação é usada pela segunda vez, logo que termina a fase de busca dos operandos, em um desvio sobre o código de operação da instrução. Este desvio serve para levar o microprograma diretamente a micro-rotina de execução da instrução, sendo usada somente a saída 1, (fig.V-5a).

O campo do microprograma LIB DECOD informa ao controle do circuito, quando o microprograma está passando em um dos pontos descritos acima, e comanda a seleção das duas saídas. Na tabela V-5 estão as configurações possíveis do campo e o seu significado para o circuito de decodificação.

LIB DECOD		
<1>	<0>	
0	0	INATIVO
0	1	LIBERA SAÍDA 1 (FIM DA BUSCA DOS OPERANDOS)
1	0	DESVIO SOBRE MODO DESTINO
1	1	LIBERA SAÍDA 1 ou SAÍDA 2 (FIM DA BUSCA INSTRUÇÃO).

TAB V-2 - Campo LIB DECOD



A → R1(15:05)= 17000

B → R1(15:05)= 00000

C → R1(15:06)= 00002

Fig. V-6 - Decodificador de Instruções

A parte do circuito de decodificação do código da instrução que gera a saída 1, é constituido de memórias que operam fazendo um mapeamento deste código que tem tamanho variável, em outro de 7 bits mais apropriado ao uso pela Unidade de Controle.

A decodificação do código começa logo que uma nova instrução é carregada no Registro de Instruções (RI). Neste processo, primeiro são analisados, através da memória 20P (fig. V-6), os bits 15 a 12 do RI que determinam se a instrução é ou não de dois operandos e no caso afirmativo qual é ela. Caso a instrução não seja de dois operandos (RI <14:12> = ØØØ ou 111) a decodificação continuará através de uma nova memória a ser escolhida pelos circuitos decodificadores 20P1 e 20P2. O circuito 20P1 tem seu funcionamento liberado quando RI <14:12> = 000 e 20P2 quando RI <14:12> = 111, e ambos utilizam os bits 15 e 11 do RI para escolher a nova memória.

Supondo que 20P1 liberado temos as seguintes possibilidades:

a) RI <15:11> = 0 000 0
b) R1 <15:11> = 0 000 1
c) R1 <15:11> = 1 000 0
d) R1 <15:11> = 1 000 1

Nas possibilidades b,c,d as memórias liberadas são endereçadas pelos bits 10 a 6 do RI e fornecem na saída um código que identifica a instrução. No caso a, entretanto,os bits 10 a 6 não são suficientes para identificar completamente todas as instruções do grupo, sendo necessário um novo nível de decodificação. Neste novo nível são usadas duas memórias, que através dos bits 5 a 0 do RI, identificam finalmente a instrução em questão (fig.V-6, memórias $(00)_8$ e $(02)_8$). Estas últimas memórias são usadas quando os bits 10 a 6 são iguais a $(00)_8$ e $(02)_8$.

Quando o decodificador 20P2 é liberado temos as seguintes possibilidades

a) R1 <15:11> = 0 111 0
b) R1 <15:11> = 0 111 1

Para estas possibilidades, é liberada somente

uma memória que através dos bits 11 a 7 identifica a instrução.

c) R1 <15:11> = 1 111 0

d) R1 <15:11> = 1 111 1

Estas duas configurações correspondem as instruções de ponto flutuante. Para cada configuração é liberada uma memória que usa os bits 10 a 6 para decodificar as instruções.Entretanto quando R1<15:11> = 11110 e os bits 10 a 6 são iguais a zero, uma nova memória é liberada para completar a decodificação da instrução, usando os bits 4 a Ø do RI ($F(00)_8$, fig.V-6).

O circuito que gera a outra saída do circuito de decodificação (saída 2), como vimos dá em código binário informações que especificam se a instrução é de um ou dois operandos bem como os modos de endereçamento usados para buscá-los. As informações sobre os modos de endereçamento são dadas em 3 dos 5 bits do código e dependem do tipo da instrução. Se a instrução é de um operando e modo destino diferente de zero,estes 3 bits são cópia dos bits do RI que especificam o modo de endereçamento destino. É interessante observar que uma instrução de um operando e modo destino zero, para efeito de desvio, se comporta como instrução imediata, já que não há busca de operandos. Estas instruções não usam a saída 2. Nas instruções de dois operandos, quando o modo fonte é diferente de zero, os 3 bits do código são copiados dos bits do RI que especificam o modo fonte. Quando o modo fonte é zero, os bits do código são tirados dos bits do RI que especificam o modo destino.Na tabela V-3 estão resumidas as várias fontes dos bits relativos aos modos de endereçamento gerados na saída 2 e as condições em que ocorrem.

INSTRUÇÃO DE 1 OPERANDO		INSTRUÇÃO DE 2 OPERANDOS		
MODO OESTINO	SAÍDA 2	MODOS		
		FONTE	DESTINO	SAÍDA 2
= Ø	USAR SAÍDA 1	= 0	= 0	USAR SAÍDA 1
# Ø	RI<02:00>	= 0	≠ 0	RI<02:00>
		≠ 0	= 0	RI<11:09>
		≠ 0	≠ 0	R1<11:09>

TAB. V-3 - Geração dos bits MODO END da Saída 2

A lógica de decodificação, em virtude do número de testes e do número de bits do endereço base que pode modifi

car, é um instrumento de desvios poderoso, que acelera enormemente a microprogramação. Se observarmos que todas as instruções tiram vantagem desta lógica para fazer desvios, já que a fase final de busca da instrução onde a lógica é usada pela primeira vez, é ponto de passagem obrigatória de todas as instruções, poderemos ter uma idéia de sua importância para o esforço de diminuir o tempo de execução das instruções.

5- Registro de Endereço da Memória de Controle

O Registro de Endereço é normalmente carregado com o resultado obtido na saída da lógica de seleção de endereços. Nesta lógica o endereço base é alterado sob controle do microprograma com sinais vindos das lógicas de desvios e decodificação. Entretanto sob certas condições o conteúdo do registro é carregado com o valor $(000)_8$. Nesta palavra da memória de controle começa uma micro-rotina que determina a condição que ocorreu, para então dar início a uma rotina que atenderá a ocorrência. As condições que podem forçar este tipo de carga no Registro de Endereço são as seguintes:

a) Sequência de inicialização, quando o processador é ligado ou quando é apertada a tecla "START" do painel com o processador em "HALT".

b) Todos os abortos que podem ocorrer durante o processamento.

6- Circuito de Sincronização

A consideração básica sobre a sincronização de microinstruções é o número de ciclos do relógio central que uma microinstrução permanece ativa. Máquinas monofásicas são aquelas em que a microinstrução permanece ativa somente durante um ciclo de relógio. Mais especificamente são máquinas em que todos os controles da palavra de microprograma são emitidos simultaneamente. Nas máquinas polifásicas os controles são emitidos sequencialmente, comandados por uma série de ciclos do relógio.

As máquinas estritamente monofásicas, na prática revelam-se muito pouco eficientes, quase que impraticáveis. Isto porque um grande número de microinstruções seria necessário para implementar as instruções, resultando em diversos acessos a memória de controle e portanto aumentando o tempo de execução. Por outro lado máquinas polifásicas poderiam resultar em palavras de microprograma muito largas, particularmente se a microinstrução permanece ativa durante um número de ciclos muito grande. Campos da palavra de controle, que tivessem de atuar mais de uma vez durante o tempo de execução da microinstrução teriam de ser repetidos, cada repetição associada com um tempo diferente. Na maioria das máquinas, as microinstruções tem um número fixo de ciclos, sendo possível em alguns casos incluir na própria microinstrução informações sobre o tempo de duração. Usualmente uma boa escolha entre os dois sistemas é projetar máquinas polifásicas, com um número de ciclos tal que seja evitada repetição de campos de controle.

As microinstruções da Unidade de Controle são do tempo polifásico, executadas em três ciclos de 40 ns, chamados T1, T2 e T3. É possível ainda ser acrescentado sob controle do microprograma, entre os tempos T2 e T3, um tempo extra de 40 ns chamado TF. A figura V-7 ilustra os tempos principais e os tempos secundários, utilizados durante a execução das microinstruções. Na figura também há um diagrama de tempos da execução de uma micro-operação básica, para mostrar a relação entre os tempos gerados e o fluxo de dados dentro da Unidade Aritmética. No início da microinstrução, em T1, o Registro de Microprograma é carregado com a palavra lida da Memória de Controle. Durante a porção do ciclo imediatamente após esta carga, os registros que serão operados, são liberados nas duas barras de comunicação internas que levam até as entradas A e B da Unidade Aritmética e Lógica (UAL). Durante a porção central do ciclo a informação propaga-se através da UAL, onde é realizada a operação e atinge a barra que leva o resultado de volta aos registros. Em T3S, ao final do ciclo, o resultado é armazenado no(s) registro(s) especificado(s) pela palavra de controle.

Para gerar os tempos é usado um circuito contador em anel. Este circuito sob certas condições pode ser parado nos tempos T1 e T2. Ao cessar a(s) causa(s) que paravam o cir

cuito, ele volta a funcionar normalmente.

Há 4 condições externas ao circuito que podem paralizá-lo em T1. Se qualquer uma dessas acontece T1 não é gerado e T3 permanece em um. O circuito contador permanece em T3 até que todas as causas que o paravam tenham sido resolvidas, neste instante T3 volta a zero e T1 vai a um. As condições que param o circuito em T1 são as seguintes:

a) Parada a cada ciclo de barra. Quando o processador é colocado em "HALT" e no modo "Parada a cada Ciclo" através do painel, ocorre a parada em T1 sempre que é completada uma entrada e saída e o bit PARE do microprograma é ativado. O processador volta a funcionar quando é apertada a tecla CONTINUE.

b) Interrupção - No final da execução de cada instrução é feito um teste pelo microprograma para descobrir se há pedidos de interrupção ou "traps" a serem atendidos. Caso haja algum pedido com prioridade bastante para ser atendido, o periférico deve enviar o endereço da rotina que trata a sua interrupção (Vetor de Interrupção). Durante a microinstrução que lê este vetor, o bit do microprograma PARE INTR é ativado, parando o contador em anel. O circuito pode ser liberado de duas maneiras diferentes após o dispositivo ter ganho o controle da barra com um pedido de interrupção. Normalmente ocorre a liberação quando o dispositivo avisa que já mandou o vetor. Entretanto também pode ocorrer o que é chamado de liberação passiva, isto é, o dispositivo faz várias transferências na barra e em seguida a libera sem enviar o vetor de interrupção.

c) Execução das Macros - As macroinstruções são executadas através de um circuito que comanda a UA independentemente da UC, realizando com rapidez tarefas repetitivas que dispenderiam muito tempo se executadas por microprograma (Ver capítulo IV-10). Durante a execução das macroinstruções, o contador em anel deve estar parado, permitindo que o circuito das macros exerça total controle sobre a UA. O gerador de tempos é interrompido utilizan

do-se sinais de controle do campo que define a macroinstrução a ser executada. Ao fim da execução o circuito das macros envia um sinal que é usado para liberar o funcionamento da UC.

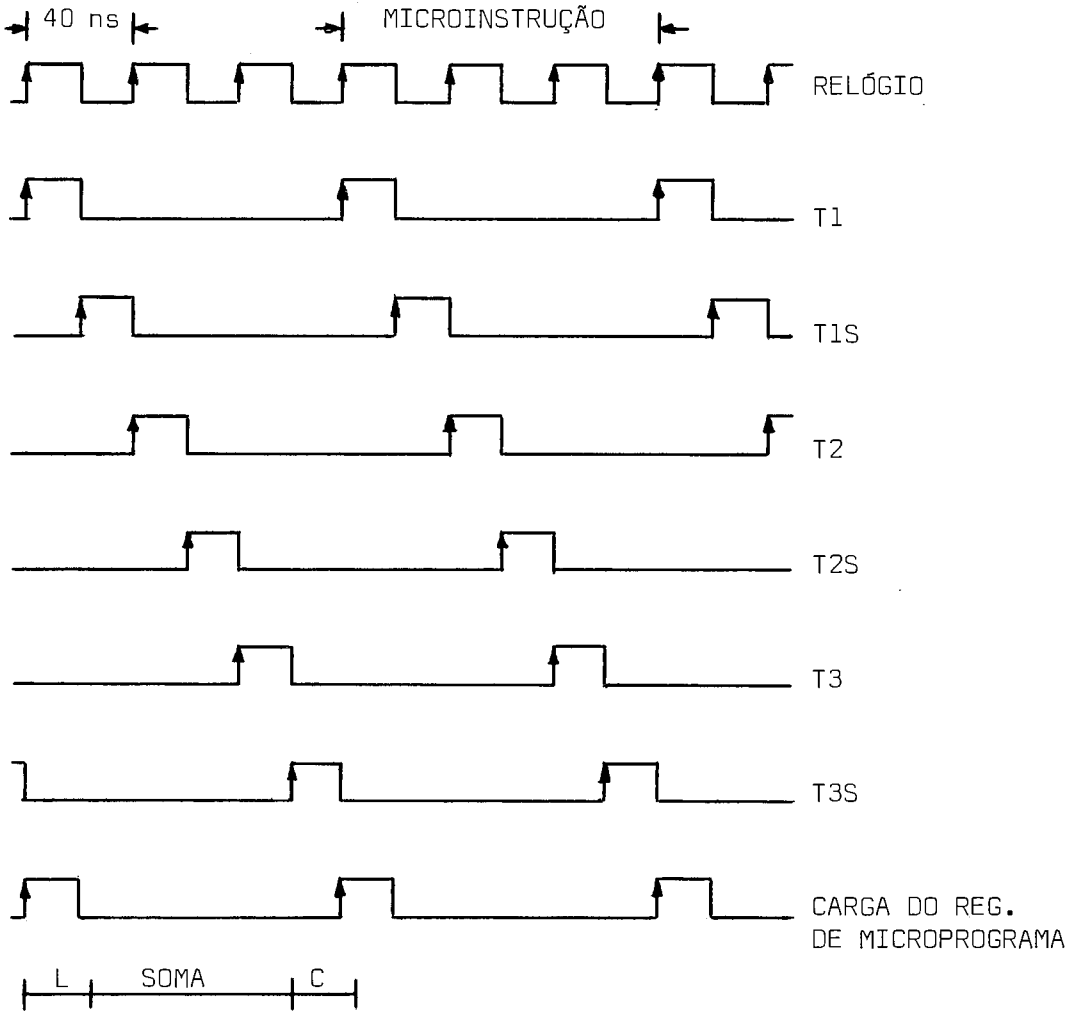
d- Operações de Entrada e Saída - Como já foi dito (Cap. V-3) as operações de E/S na UCP são realizadas em paralelo com a execução dos microprogramas,visando diminuir tempo de execução das instruções. Quando é necessária uma operação de entrada e saída, o microprograma envia para a Unidade de E/S sinais com informações sobre o tipo de operação a ser realizada. Estes sinais especificam se o acesso será feito a uma instrução, índice ou operando e se ocorrerá uma leitura ou escrita. Um sinal também do microprograma valida estas informações avisando a UE/S que deve ser iniciada a operação. O microprograma, no caso de leitura avança até que o dado seja indispensável ao prosseguimento da micro-rotina e aí para, ou no caso de escrita avança até que seja necessário nova operação de E/S, quando ocorre a parada.
O circuito é parado através do bit do microprograma PARE. Quando a E/S termina,um sinal é enviado pela UE/S avisando que completou a tarefa, liberando o funcionamento do contador. No caso da E/S terminar antes que seja ativado o bit PARE, o microprograma não chega a ser interrompido.

Também é possível parar o circuito que gera os tempos em T2, como consequência de uma das duas ações que mostramos a seguir:

a) Parada Micro a Micro - Em depuração e manutenção é importante acompanhar o funcionamento dos microprogramas passo a passo. Desta maneira é possível acompanhar as várias ações que são executadas durante a microinstrução e verificar algum mau funcionamento. Para permitir esta facilidade, há chaves no painel de manutenção que permitem que o microprograma seja executado microinstrução a microinstrução. Cada vez que a tecla CONTINUE é apertada após a parada, uma nova microinstrução é execu

tada.

b) Parada em um Endereço Pré-fixado - Uma outra forma de fazer o microprograma parar em T2 e que também serve para auxílio a depuração, é utilizar o Registro de Parada de Microprograma. O conteúdo deste registro serve para fixar um endereço qualquer da memória de controle como ponto onde o microprograma deve parar ao passar. Este registro, aparece ao programador como uma posição da memória principal e portanto pode ser carregado por instruções da máquina. Depois da Unidade de Controle parada, a fim de reiniciar a execução deve ser usada a chave CONTINUE.



L - LIBERAÇÃO

C - CARGA

Fig. V-7 - Diagrama de tempos da Unidade de Controle

VI - MICROPROGRAMAÇÃO DA UNIDADE DE CONTROLE

1- Introdução

Na UCP, a microprogramação não foi utilizada com a finalidade de dar ao programador a possibilidade de alterar o conjunto básico de instruções, para adaptá-lo à sua aplicação particular. Ao contrário, ela foi empregada para que um conjunto fixo de instruções fosse implementado de maneira eficiente.Deste modo a microprogramação nesta máquina é área de trabalho exclusiva dos projetistas. A flexibilidade é transmitida ao programador na forma de um conjunto de instruções e modos de endereçamento eficientes.

As funções dos microprogramas na UCP são as seguintes:

a) Buscar as instruções da máquina e os dados localizados na memória.

b) Controlar as sequências de execução das instruções.

c) Executar as funções de painel.

d) Em interrupções, "traps" ou erros, salvar o Contador de Programa (CP) e a Palavra de Estado do Processador (PEP) e buscar um novo par CP, PEP.

Quando da criação dos microprogramas, as instruções foram agrupadas de modo que na execução de funções comuns as diversas instruções fossem utilizadas as mesmas micro-rotinas. Por exemplo, em algumas instruções as fases de cálculo de endereço e busca dos operandos são realizadas para todo o grupo pelas mesmas micro-rotinas e somente na fase de execução é que há separação por instrução. Este processo lembra o conceito de subrotina, mas algumas características são diferentes. Na UC não existe microinstrução de chamada de subrotina, a entrada em execução de qualquer micro-rotina é feita através de desvios condicionais. Ao fim da execução não existe um endereço de retorno, a escolha do caminho a seguir é feita também por desvios condicionais.

Para efeito da microprogramação as instruções fo

ram primeiramente divididas em três grandes grupos, segundo o número de operandos:

a) Instruções de execução imediata.

b) Instruções de um operando.

c) Instruções de dois operandos.

Reunindo-se as tarefas necessárias para executar estas instruções, chegamos a estrutura básica para os microprogramas, que está mostrada na figura VI-1.

A execução das instruções é iniciada pela micro rotina de busca e decodificação, que obviamente é comum a todas. Nesta micro-rotina é feito um teste para verificar a ocorrência de eventos assíncronos, tais como: interrupções, "traps" e pedidos de painel. No final do processo de decodificação é realizado outro teste para descobrir a qual dos três grupos pertence a instrução.

Caso a instrução seja de execução imediata, este desvio leva diretamente a micro-rotina de execução da instrução em questão. No caso de haver necessidade de busca de operandos, há duas possibilidades; em instruções de dois operandos, primeiro é feito a busca do operando fonte, passando-se em seguida a busca do segundo operando ou operando destino; em instruções de um operando é imediatamente iniciada a busca do operando destino, com a mesma micro-rotina utilizada pelas instruções de dois operandos. No final da busca do último operando há um desvio sobre o código de operação, que levará diretamente a micro-rotina de execução da instrução em questão.

A seguir vamos dar algumas informações sobre estas micro-rotinas, mostrando as principais tarefas que elas realizam, sem entrar em detalhes de implementação. Maiores informações, podem ser encontradas nos manuais de descrição da UCP[10] e nos fluxogramas das microinstruções[11].

2- Micro-Rotina de Busca e Decodificação das Instruções

A execução de todas as instruções é iniciada através desta micro-rotina, que além das duas funções básicas rea

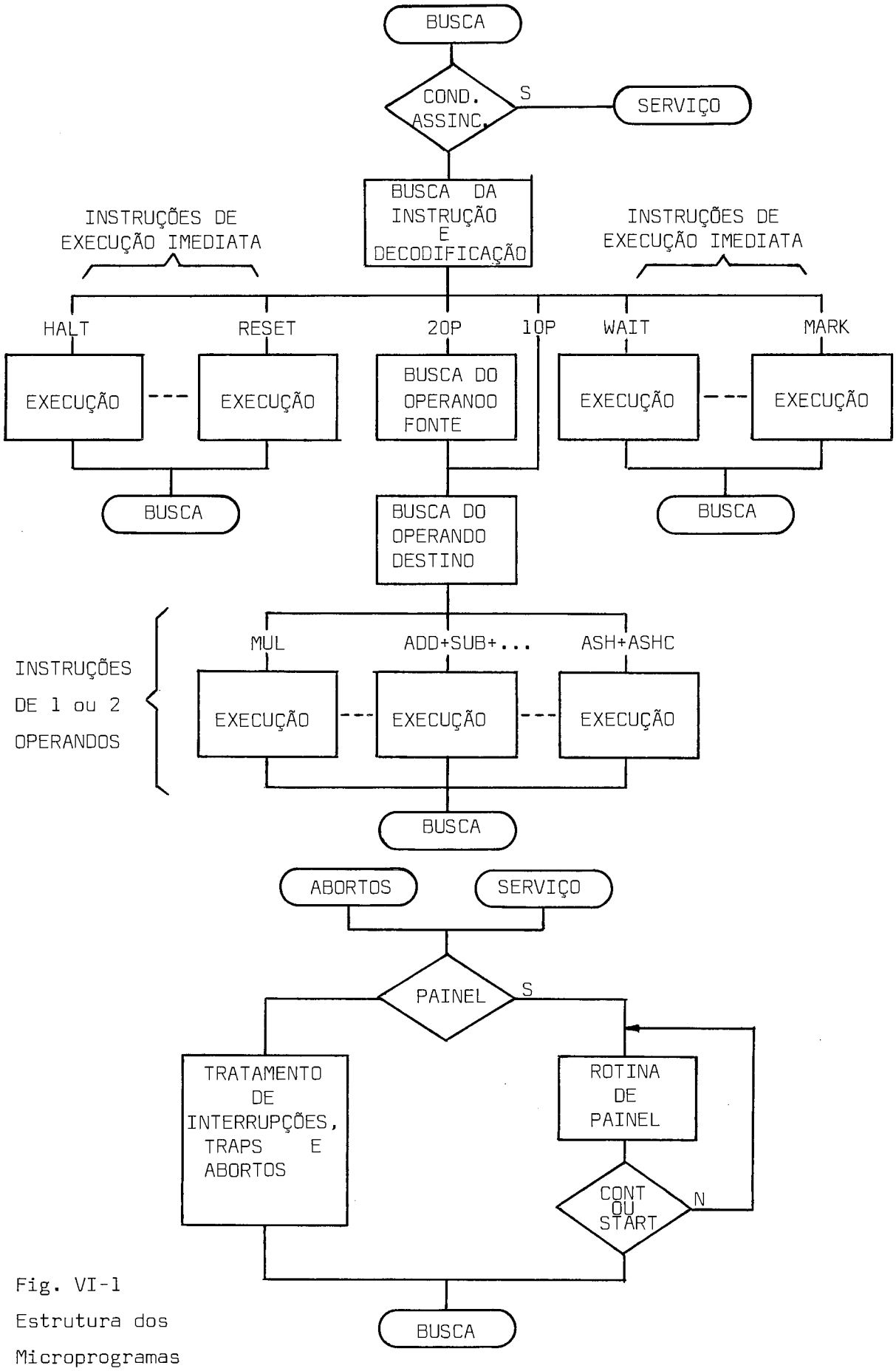


Fig. VI-1
Estrutura dos
Microprogramas

liza outras tarefas. O registro Contador de Programa (CP) deve ser atualizado de modo que fique apontando a próxima instrução a ser executada. Após a instrução ter sido lida da memória e carregada no Registro de Instruções (RI), os registros gerais apontados pelos campos de endereçamento fonte e destino devem ser lidos da memória rascunho e carregados nos registros de trabalho da UAL. É interessante observar que apesar da decodificação da instrução ser completada somente no final da execução da micro-rotina, os registros gerais são lidos como se a instrução fosse de um ou dois operandos. Isto é feito para adiantar possíveis tarefas futuras, já que durante a decodificação o microprograma não teria nenhuma a realizar. Outra importante tarefa realizada antes da busca da instrução ser completada é a verificação da ocorrência de eventos assíncronos. Esta verificação é feita através de teste sobre um sinal enviado pela UE/S que informa da ocorrência ou não de tais eventos. No caso afirmativo há um desvio para a micro-rotina de tratamento de erros,interrupções,etc. Ao final da execução da micro-rotina de busca é realizado um desvio sobre o código de operação da instrução, para então ser iniciada a busca dos operandos ou execução da instrução, conforme o caso.

3- Micro-Rotina de Busca do Operando Fonte

Esta micro-rotina é usada pelas instruções de dois operandos para busca do primeiro operando. Ela tem quatro pontos de entrada diferentes, utilizados conforme o modo de endereçamento especificado pela instrução. Um dos pontos é utilizado no modo 1 e os outros três para instruções com modos 2 e 3,4 e 5, 6 e 7. Nesta micro-rotina é assumido que o conteúdo do registro geral a ser utilizado no cálculo do endereço, já está armazenado nos registros B e EA da UA, tarefa realizada na micro-rotina anterior. Todos os registros gerais alterados no processo de busca do operando (operações de auto-incremento e decremento), também são modificados na memória rascunho. Ao final da execução desta micro-rotina quando é feito um desvio para determinar qual a próxima a ser executada, o operando fonte está sendo esperado no registro de Barra (RB). Foram utilizadas para implementar a busca

do operando fonte nos sete modos de endereçamento, 11 microinstruções, o que dá uma idéia da maneira compactada com as micro-rotinas foram implementadas.

4- Micro-Rotinas de Busca do Operando Destino

A busca do operando destino é feita por 7 micro-rotinas independentes, uma para cada modo de endereçamento, exceto nos modos 6 e 7 em que a micro-rotina é em grande parte comum aos dois modos. As instruções que utilizam estas micro-rotinas (instruções de 1 e 2 operandos), foram divididas em três grupos, para acelerar a execução das instruções.

- Grupo M (movimentação) - MOV e MTP
- Grupo J (desvio) - JMP e JSR
- As instruções restantes.

Com esta divisão as micro-rotinas passaram a ter duas fases. A primeira fase é comum a todas as instruções e consiste no cálculo do endereço do operando. No instante em que a UC tem o endereço pronto, há três procedimentos diferentes a seguir segundo o grupo da instrução. Nas instruções que não pertencem aos grupos J ou M, deve ser iniciada a busca do operando destino. Em instruções do grupo M, esta busca não é necessária, sendo imediatamente iniciada a escrita do operando fonte no lugar apontado pelo endereço destino. Enquanto que nas instruções do grupo J, simplesmente é feita a carga do endereço no Contador de Programas, para que a próxima busca de instrução seja feita neste endereço, concretizando-se o desvio. Com a divisão em três grupos é poupado a estes dois últimos a fase de busca do operando destino, que lhes é inútil.

5- Micro-Rotina de Serviço

Esta micro-rotina é usada durante o tratamento de todos os abortos, "traps" e interrupções.

Um aborto é a interrupção de uma entrada e saída devida a um erro. Os abortos são atendidos no instante em que

ocorrem, fazendo com que a execução da instrução seja interrompida.

Um "trap" é a interrupção de um programa por condições internas a máquina. Estas condições podem ser erros ou não. Por exemplo, um "halt" de painel provoca um "trap".

Uma interrrupção é semelhante a um "trap", mas é causada por condições externas a máquina. Estas condições podem ser provocadas por programas (interrupção por programação) ou por periféricos. Tanto interrupções como "traps" são atendidos ao final da execução das instruções.

Quando da ocorrência de um "trap" ou interrupção, o microprograma é desviado para a micro-rotina de serviço por meios normais, ou seja, um teste que é realizado durante a execução da busca da instrução. Entretanto em abortos, o endereço inicial da micro-rotina é forçado por circuito, no registro de endereços da memória de controle, alterando o fluxo normal dos dos microprogramas. Podemos dizer que nas ocorrências de aborto de programa, o microprograma também é abortado, sendo desviado diretamente para a micro-rotina de serviço. Por problemas de sincronismo, a carga do endereço não é feita exatamente no instante em que a UE/S sinaliza informando que ocorreu um erro na operação de E/S. Há uma espera até a UC pare o microprograma, aguardando informações de que a E/S foi completada. Assim é garantido que a UC neste instante não está executando nenhuma microordem, não havendo perigo de, por exemplo, ser interrompida pela metade uma escrita em um registro geral.

Durante a execução desta micro-rotina, o par Contador de Programa (CP), Palavra de Estado do Processador (PEP) da subrotina que é requerida pelo aborto, "trap" ou interrupção, é lido da memória e em seguida o par CP, PEP da instrução em execução é salvo na pilha determinada pelos bits 15 e 14 da nova PEP.

O endereço da memória onde está armazenado o CP da subrotina é denominado vetor. No endereço igual a vetor mais dois é armazenado a PEP. Durante as interrupções este vetor é fornecido pelo dispositivo que interrompe e lido durante esta micro-rotina. Em abortos e "traps" o vetor é gerado internamente ao processador, e também lido pela micro-rotina.

6- Micro-Rotina de Painel

As funções do painel são executadas por microprograma. A micro-rotina de painel é executada sempre que a tecla "HALT" é apertada indicando que o operador deseja realizar alguma tarefa pelo painel. Esta micro-rotina opera basicamente como uma rotina de teste de estado. O microprograma fica em laço testando sinais enviados pelo painel, que informam quando uma tecla foi apertada, e qual foi ela. Quando o operador pressiona alguma tecla o microprograma realiza um desvio para a micro-rotina apropriada, voltando ao final da execução da função para o laço de teste de estado. É interessante observar que estes sinais enviados pelo painel são usados diretamente, no instante do desvio como parte do endereço da próxima microinstrução a ser executada. Deste modo o desvio é feito em um passo, não sendo necessário testes sucessivos para atingir a micro-rotina desejada. Para o microprograma sair da micro-rotina de painel é suficiente que o operador aperte a tecla CONTINUE ou a tecla "START" com a de "HALT" desligada.

7- Micro-Rotina de Execução

As instruções na fase de execução também foram agrupadas, agora segundo o tipo de operação que deve ser realizada, de modo que pudessem ser executadas pela mesma micro-rotina. Em alguns casos foram necessários acréscimos nos circuitos da Unidade Aritmética para tornar possível este agrupamento.

Para a execução de diversas instruções aritméticas e lógicas (ver tabela VI-1), como já foi citado (Cap. V-2), foi criada uma lógica adicional para controle da Unidade Aritmética e Lógica (UAL). Ela normalmente é controlada por um campo do microprograma, que especifica a função a ser executada durante a microinstrução. Entretanto quando alguma instrução deste grupo está em execução, a microinstrução transfere o controle da UAL para uma memória de leitura exclusiva. Esta memória recebe bits do Registro de Instruções, suficientes para identificar a instrução e fornece na saída sinais para controle da UAL.

Da mesma forma o posicionamento dos códigos de

condição para estas instruções, teve de ser feito condicionalmente, com o mesmo processo usado no controle da UAL. Normalmente o microprograma escolhe os pontos da máquina (saída da UAL vai um, sinal do resultado, etc.), que serão usados neste posicionamento. Entretanto, como nesta micro-rotina, a UC desconhece a instrução em execução, esta escolha tem de ser feita automaticamente. Para isto, também foi usada uma memória, cuja função é decodificar a instrução e fornecer na saída sinais que escolherão os pontos utilizados no posicionamento dos códigos.

Com estas modificações as terefas da microinstrução que executa as instruções do grupo são: liberar os registros (RB e B) onde estão os operandos nas barras de entrada da UAL, transferir os controles da UAL e circuitos que posicionam codigos de condição para as memórias auxiliares e ao final da operação, carregar o resultado no registro pré determinado (RB).

GRUPO E1	GRUPO E2	GRUPO T
CLR.B	BIC.B	CMP.B
COM.B	BIS.B	BIT.B
INC.B	ADD	TST.B
DEC.B	SUB	
ADC.B		
SBC.B		
XOR		

Tabela VI-1

Outras instruções também foram agrupadas (Ver Cap. IV-6), para que fosse possível economizar memória de controle. Maiores informações sobre as micro-rotinas de execução são encontradas no manual de descrição do processador[10].

VII - CONSIDERAÇÕES SOBRE A IMPLEMENTAÇÃO DO PROJETO

Neste capítulo é apresentada uma descrição sumária das tarefas realizadas durante o projeto da Unidade de Controle e uma previsão das tarefas ainda a realizar. São abordados os problemas enfrentados em cada uma das etapas e mostrada as soluções adotadas.

A implementação da UC e também das demais partes que compões a UCP (Unidade Aritmética, Sistema de Entrada e Saída, Sistema de Memória e Painel), pode ser dividida em várias etapas, que são comuns a cada uma delas. A seguir damos uma descrição de como o projeto se desenvolve em cada etapa.

1a. Etapa - Definição da Arquitetura

O projeto da UCP foi iniciado no final de 1975, embora a idéia tenha sido objeto de discussão durante todo aquele ano. Nesta fase uma proposta inicial para a organização interna foi discutida entre os 12 membros da equipe de desenvolvimento de circuitos, então envolvidos no projeto. A discussão visava adaptar a proposta ao conjunto de instruções da máquina. Esta adaptação significava principalmente verificar a possibilidade de executar o conjunto de instruções, usando os recursos disponíveis na proposta e procurar otimizá-la de maneira a conseguir redução nos tempos de execução da instrução. Das observações realizadas, foram sugeridas modificações nos caminhos de dados, alterações na localização de registros, etc.

Uma vez definida a organização interna básica, o projeto foi dividido em 5 partes, entregues cada uma delas à responsabilidade de um dos membros da equipe. Estas partes que refletem a maneira usual de dividir as UCPs estão a seguir:

a) Unidade Aritmética, circuitos de macroinstruções e microprogramas de instruções flutuantes.

b) Unidades de Controle, microprogramas das instruções inteiras e de tratamento de erros e de painel.

c) Unidade de Entrada e Saída, registros internos e Sistema de Relocação da Barra de Periféricos[8].

d) Sistema de Memória (Memória tipo "Cache-Backing")[9].

e) Painel, Carregador de Microprogramas e Depurador Programável de Circuitos Digitais[13].

Os trabalhos desenvolvidos em cada uma das partes com excessão da Unidade Aritmética, deveriam resultar em teses, algumas já apresentadas a COPPE pelos responsáveis, com o objetivo de obter o grau de mestrado. O Depurador Programável e o Carregador de Microprogramas foram incluidos com a finalidade de servir como ferramentas de depuração do projeto.

2a. Etapa - Projeto Lógico

O desenvolvimento desta etapa foi bastante prejudicado com a redução do pessoal envolvido. Esta redução ocorrida durante o ano de 1976, se deveu principalmente a mudança de prioridade do projeto dentro do NCE e ao afastamento de alguns elementos da instituição. No final de 1976 a equipe de projetistas da UCP estava reduzida a quatro engenheiros. Estes fatos não só tiveram como consequência um aumento no tempo previsto para o projeto lógico, como se refletirão na fase de montagem e depuração da UCP. Um primeiro problema salta aos olhos, um projeto que foi inicialmente separado em cinco partes, não poderia ser levado com rapidez por quatro engenheiros. Segundo é de consenso geral na equipe que seria mais proveitoso em ternos de rapidez de implementação e de idéias para o projeto, que houvesse pelo menos dois projetistas em cada parte. Somente em 1978 passamos a contar com a contribuição de mais um projetista, de um coordenador administrativo e de três estagiários de engenharia. Atualmen integram a equipe, um coordenador, sete projetistas e três estagiários.

A tarefa principal desta fase consiste em, a partir da definição das funções da Unidade de Controle, implementar circuitos lógicos que realizem estas funções. Nesta implementação foram usados circuitos integrados convencionais da linha TTL. Devido aos requisitos de velocidade da máquina, a quase totalidade dos integrados pertence a linha TTL Schottky, que tem quando

comparada a linha normal, velocidades de propagação duas a três vezes maior.

A microprogramação das instruções caminhou em paralelo com a etapa de projeto lógico das diversas unidades possibilitando realimentações que serviram para apurar a organização interna da máquina. A partir de necessidades descobertas durante a microprogramação, houve alterações no projeto, que levaram a diminuição do tempo de execução das instruções, um objetivo constante durante todo o projeto.

A documentação gerada nesta etapa constou de um conjunto de desenhos dos circuitos lógicos, um conjunto da fluxogramas dos microprogramas e um manual com descrição do funcionamento dos circuitos. Esta fase durou de janeiro de 1976 até outubro de 1978.

3a. Etapa - Partição do Projeto e Elaboração da Documentação para Montagem.

O trabalho de partição engloba um conjunto de tarefas preparatórias, necessárias para a montagem do projeto em placas, que serão do tipo CAMBIOM, com capacidade para 108 integrados e 140 pinos de conector.

A primeira tarefa é dividir o projeto em módulos que devem preferencialmente constituir circuitos com funções definidas dentro da UC. Em seguida estes módulos são agrupados,procurando juntar aqueles que mais trocam sinais entre si, com o objetivo de colocar os grupos de módulos em placas. As limitações que devem ser observadas neste processo, referem-se ao número de circuitos integrados que a placa pode abrigar e ao número de pinos de conectores disponíveis. É interessante observar que na maioria dos casos a maior limitação foi o número de conectores,o que obrigou a reunir em placas módulos que não eram afins,ou mesmo fazer divisão de módulos, embora fosse melhor mantê-los unidos.

Após a partição, o projeto da UC, que utiliza em torno de 340 pastilhas de circuitos integrados ficou dividido em quatro placas, da seguinte forma:

a) Placa TIM - Circuitos para gerar os pulsos de sincronis

mo.

b) Placa DCC - Circuitos para decodificação das instruções e posicionamento dos códigos de condição da Palavra de Estado do Processador.

c) Placa RMA - Parte da Memória de Controle.

d) Placa RMB - Restante da Memória de Controle e Circuitos de Endereçamento da Memória de Controle.

O passo seguinte é a preparação das listagens para a montagem dos circuitos. Estas listagens, que são obtidas através de um programa desenvolvido no NCE/UFRJ, dão a relação de todas as ligações que devem ser feitas entre os componentes existentes na placa e entre eles e os conectores, através de caminhos mínimos. Como resultados adicionais são obtidos os comprimentos dos fios necessários as ligações e os pinos utilizados em cada circuito integrado e nos conectores.

Para este programa, antes devem ser preparados os seguintes dados: posição que os diversos componentes ocupam nas placas, pinos do conector utilizados e uma relação das ligações a serem realizadas.

A partição do projeto da Unidade de Controle foi realizada durante o mês de abril de 1979 por Armando Drumond e Paulo Henrique de Aguiar Rodrigues, dois integrantes da equipe.

A documentação gerada para cada placa, durante a partição e que será utilizada durante a fase de montagem consiste de:

- Um diagrama com a especificação da posição que os componentes ocupam nas placas.
- Uma relação dos pinos dos conectores utilizados com os nomes dos sinais e as suas características de carga, isto é, quanto de corrente é fornecido ou necessário receber.

4a. Etapa - Montagem e Elaboração das Rotinas de Teste.

Nesta etapa adotou-se a norma de somente iniciar

a montagem das placas após a elaboração de uma rotina para o teste de cada uma delas na bancada. A função principal desta rotina é simplificar o trabalho de depuração das placas,tornando os procedimentos para os testes mais automáticos. Deste modo uma placa ao chegar à bancada para ser depurada, já tem estabelecido todos os pontos onde devem ser aplicados sinais e os pontos onde devem ser lidos os resultados e que valores serão lidos. A idéia de primeiro desenvolver as rotinas vem da experiência adquirida durante a depuração do Sistema de Entrada e Saída. Foi observado que durante o exame dos circuitos para a elaboração das rotinas, alguns erros são detetados, sendo então mais facil a correção destes erros quando tudo está em papel do que após a placa já montada.

Nas rotinas são especificados equipamentos de teste existentes no NCE/UFRJ, entre eles alguns projetados na própria instituição. Um depurador não inteligente de circuitos digitais, desenvolvido no NCE, é o equipamento base para a preparação dos testes, permitindo:

- Aplicação de sinais com tensão correspondente aos níveis lógicos "1" ou "Ø" da família TTL.
- Leitura de saídas dos circuitos por meio de díodos emissores de luz (DEL).
- Aplicação de pulsos positivos e negativos.
- Leitura de transições negativas ou positivas ocorridas em pontos do circuito sob teste.

A montagem dos circuitos como já vimos será feita em placas do tipo CAMBIOM grande, usando-se a técnica "wire-wrap". Todos os circuitos integrados antes de serem colocados nas placas são testados por meio do Testador de Circuitos Integrados desenvolvido no NCE/UFRJ[12].

5a. Etapa - Depuração das Placas

Esta depuração é realizada para cada placa isoladamente segundo os procedimentos descritos nas rotinas de teste. No caso de ser encontrado algum erro, foi estabelecido que além

da correção na placa, toda a documentação tem de ser corrigida imediatamente, isto para evitar sua desatualização. Deste modo é possível manter um mínimo de organização, fator indispensável devido ao tamanho e dificuldade do projeto, para o seu andamento com eficiência.

Deve ser observado que nesta etapa ainda não serão depurados os microprogramas. Os testes referem-se somente aos circuitos lógicos, e mesmo neste caso, estes testes, não serão realizados na velocidade de operação da máquina.

6a. Etapa - Depuração de Conjunto das Placas

Nesta etapa, primeiro será feita a depuração conjunta de todos os circuitos que compõem a UC e em seguida dos microprogramas. Na depuração dos microprogramas, inicialmente serão testadas independentemente todas as micro-rotinas sem preocupação da execução de nenhuma instrução. A seguir haverá a ligação das micro-rotinas e todo o conjunto será testado executando as diversas instruções.

Cabem aqui algumas observações sobre o processo de depuração dos microprogramas. Como já dissemos (Cap.V-2), a memória de controle será implementada com pastilhas MLE programáveis de 512 palavras de 4 bits. Isto obriga que os microprogramas sejam depurados antes que as memórias sejam programadas, evitando-se deste modo perda de pastilhas por alterações surgidas durante os testes. Normalmente esta depuração é parcialmente executada em simuladores, realizando-se somente os testes finais em protótipos. Devemos observar que mesmo com o auxílio de simuladores, o problema da depuração não é totalmente resolvido, isto porque o método não consegue reproduzir todos os aspectos da máquina, principalmente os relativos aos sincronismos ao nível dos circuitos. No projeto da UCP onde houve sempre preocupação em tornar a máquina mais veloz possível, este problema é bem mais complexo.

Embora fosse conveniente, não foi possível à equipe projetar um simulador. Por esta razão os testes dos microprogramas serão realizados na própria máquina, que sofreu então algumas adaptações. A principal delas é a utilização de uma memó

ria de acesso randômico (MAR), como unidade armazenadora dos microprogramas durante os testes, o que permite alterá-los com maior facilidade. Surge então a necessidade de projetar um sistema de carga dos microprogramas, que seja prático e flexível de modo que as modificações que se fizerem, sejam prontamente testadas.A descrição deste sistema pode ser vista com mais detalhes no trabalho de tese de Rogério Antonio Sampaio Parente Vianna[13]. Mais adiante descreveremos as modificações feitas na UC para possibilitar a utilização das MARs e alguns detalhes do Sistema Carregador, que facilitam a depuração.

7a. Etapa - Teste da UCP e Adaptação do "Software"

Após o término dos testes de cada parte isoladamente, serão iniciados os testes de funcionamento conjunto das partes. Inicialmente serão executados programas carregados na memória através do painel, com a UCP funcionando a uma velocidade baixa. Progressivamente procurar-se-á atingir a velocidade desejada. Somente após estes testes será verificado o comportamento da UCP diante de condições de erro e das ligações com periféricos.

A seguir será finalmente iniciada a fase de adaptação da máquina ao "software" da D.E.C.. Primeiro serão rodados Sistemas Operacionais mais simples procurando ajustar todos os circuitos. Somente após terem sido superados os problemas deste ajuste é que se passará a testar o funcionamento da UCP sob controle do IAS, que é o Sistema Operacional projetado para os computadores PDP 11/70.

Na execução destas duas últimas etapas, o carregador de microprogramas será o principal instrumento de depuração. Isto porque, como já dissemos antes, os microprogramas serão depurados na própria máquina, que teve então de sofrer algumas alterações. A seguir vamos descrever estas alterações e as facilidades introduzidas na UC para facilitar a sua depuração através do carregador.

A principal alteração é a utilização de MARs como unidades armazenadoras em substituição as MLEs, possibilitando que os microprogramas sejam alterados dinamicamente. Esta ca

pacidade é importante por permitir que as modificações sejam feitas com facilidade, durante o processo de depuração.' O carregador de microprogramas foi o sistema desenvolvido para dar os pessoal de depuração da UC meios de escrita e leitura na memória de controle, através de um procedimento simples e rápido, além de possibilitar o controle da própria execução desses microprogramas.

A substituição de tipos de memória não foi imediata e alguns problemas de organização tiveram de ser superados. A memória de controle tem 512 palavras de 113 bits e na sua versão final com MLEs, serão utilizadas 29 pastilhas com 512 palavras de 4 bits, e tempo de acesso no caso pior de 50 ns. Para implementar esta organização de memória com MARs, há o problema de só existirem comercialmente disponíveis, com um tempo de acesso compatível, pastilhas com 1024 palavras de 1 bit. Para contornar este problema a palavra de controle de 113 bits foi dividida em duas partes, uma de 56 outra de 57 bits. Estas partes então são armazenadas em duas palavras consecutivas de uma memória que tem 1024 palavras de 57 bits. Uma palavra de microprograma para ser lida, necessita de dois acessos seguidos a nova memória de controle.

A figura VII-1 ilustra como foi o esquema de troca das memórias.

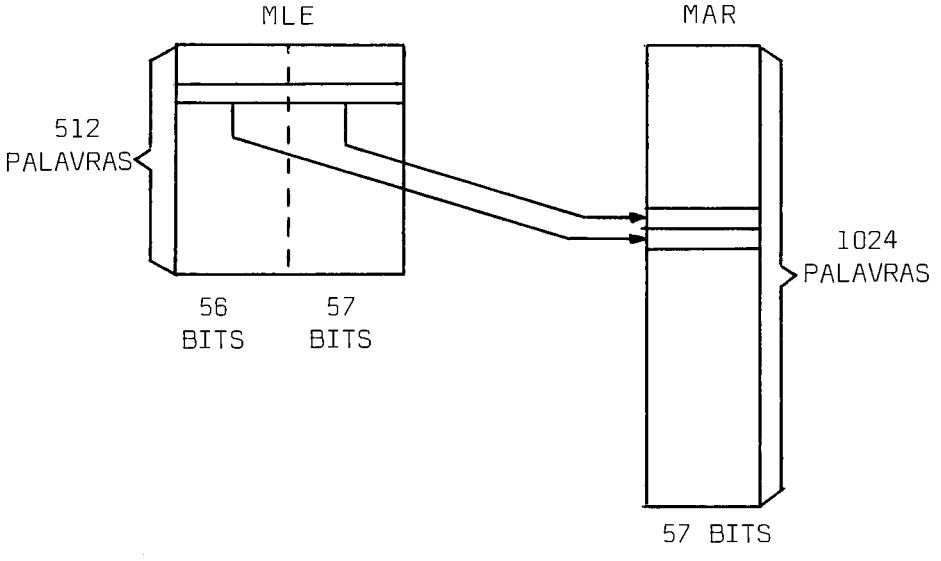
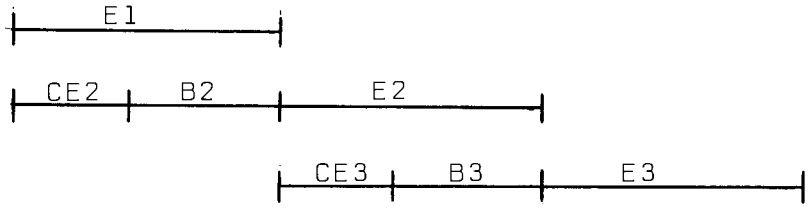


Figura VII-1 - Organização das Memórias com MAR e MLE

Superado este problema, restou ainda a dificuldade de encaixar dois acessos a uma memória de 45 ns de ciclo e mais o cálculo do endereço em 120 ns, que é o tempo esperado para a execução de uma microinstrução. Isto porque o projeto prevê que o acesso à próxima palavra a ser executada é feito em paralelo com a execução da atual, (Cap.V 4-1,método de busca em paralelo). Como isto não é possível foi adotada uma outra solução. A busca da primeira metade da palavra é feita durante o tempo de execução da microinstrução atual, o qual foi mantido em 120 ns. Quando termina a execução, o conteúdo da metade lida é armazenado em um registro, e um novo ciclo de acesso a memória é iniciado para busca da outra metade. Durante este ciclo extra, a execução dos microprogramas é interrompida. O ciclo extra dura 80 ns e somente no seu final é então iniciada a execução da nova microinstrução. Devemos notar que o tempo de execução das microinstruções não foi alterado, para fazer com que os testes reflitam ao máximo as condições de velocidade em que ocorre esta fase. É interessante comparar este esquema (fig.VII-2.b), com o esquema da versão definitiva (fig. VII-2.a).Observamos que na versão definitiva as microinstruções tem a sua fase de execução iniciada logo em seguida ao término da anterior, enquanto que no esquema para depuração haverá um intervalo entre as fases de execução, para busca da segunda metade da palavra.

a) Versão definitiva



b) Versão de Testes

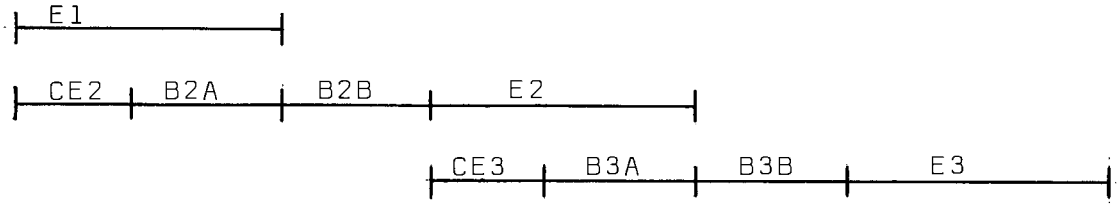


Fig. VII-2 - Temporização da Busca das Microinstruções

carregador. Dois fatores contribuiram para esta diferença:primeiro o sistema está sediado em um terminal inteligente desenvolvido no NCE/UFRJ que opera com 8 bits; segundo ele estará situado fora do bastidor da UCP e não seria prático fazerem-se ligações de no mínimo 57 fios, que é a largura da MAR, entre os dois sistemas. Para o carregador então, a palavra da memória, que como vimos representa metade da palavra de controle, foi subdividida em 7 palavras de 7 bits e uma de 8 bits (fig. VII-3), sendo os caminhos de dados reduzidos para 8 bits. Deste modo quando é necessário a leitura ou escrita de uma microinstrução são realizados 16 acessos sequenciais a memória, podendo-se concluir que ela se comporta para o carregador como se tivesse 8 K palavras de 8 bits.

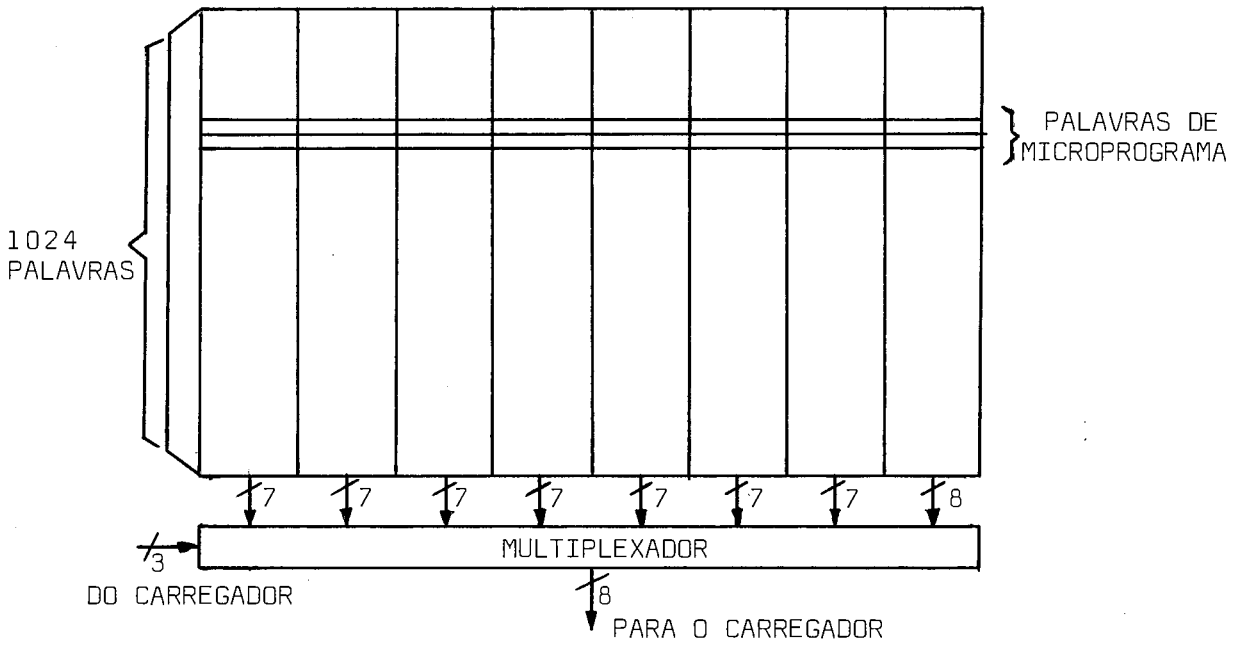


Fig. VII-3 - Memória de Controle vista do Carregador.

Tendo sido estabelecida a organização da memória de controle com MARs, foram definidos comandos que permitem ao carregador escrever e ler da memória e também escrever no seu registro de endereços. Com estes comandos foi possível construir programas no Terminal Inteligente que carregam toda a memória, alteram apenas uma microinstrução, mantem cópia atualizada da memória de controle em arquivos permanentes, etc. Todas estas facilidades são de grande importância para o processo de depuração.

Além desses comandos, foram incluidos outros pa

ra controlar o próprio funcionamento da UC:

a) Comando de Parada - Interrompe o funcionamento da UC.

b) Comando de Execução - Dá início ao funcionamento da UC, que estava parada por um dos seguintes motivos: comando de parada, UC funcionando no modo micro a micro,ou quando o endereço da palavra de controle que está em execução é igual ao conteúdo do registro de parada de microprograma.

c) Comando de Carga no Registro de Parada do Microprograma Este registro é utilizado para introduzir pontos de parada na execução dos microprogramas. Através de uma chave no painel de depuração é liberado um circuito que compara o seu conteúdo com o endereço da microinstrução que está sendo executada, quando os dois são iguais a UC é parada.

Todos esses circuitos extras, adicionados para facilitar o processo de depuração, serão retirados quando da construção de um segundo protótipo em circuito impresso e com a memória de controle implementada com pastilhas de memória de leitura exclusiva.

Acreditamos que nestes testes com o primeiro protótipo, em que todas as ligações são fiadas, não será possível atingir a velocidade desejada (Relógio Central de 25 MHZ). Portanto o principal objetivo, neste início, deve ser a depuração "lógica" do projeto, isto é, procurar fazer com que ele seja "software" compatível com o computador PDP 11/70. Numa etapa posterior, durante a construção e depuração de um segundo protótipo em circuito impresso, a máquina será otimizada no aspecto relativo a velocidade.

O projeto da UCP, excluindo a memória principal terá em torno de 1800 circuitos integrados alocados em 20 placas CAMBIOM tamanho grande, distribuidos em 2 bastidores duplos com capacidade para 13 placas cada.

VIII - BIBLIOGRAFIA

1- WILKES, M.V. - "The Best Way to Design an Automatic Calculating Machine", Manchester University Computer Inaugural Conference, p.16, julho, 1951.

2- WILKES, M.V. - "Microprogramming", Proceedings of the Eastern Joint Computer Conference, p.18, Dezembro,1958.

3- WILKES, M.V., RENWICK, W. e WEELER, D. - "The Design of a Control Unit of an Electronic Digital Computer", Proceedings of the IEEE, 105, p.121, 1958.

4- WILKES, M.V., STRINGER, J.B. - "Microprogramming and the Design of the Control Circuits in an Electronic Digital Computer", Proceedings of the Cambridge Philosophical Society, 49 part 2, p.230-238, 1953.

5- GLANTZ, H.T. - "A Note on Microprogramming", Journal of the Association for Computing Machinery, 3 nº1, p.77-78, 1956.

6- GRASSELI, A. - "The Design of Program - Modifiable Micro-Programmed Control Units", IRE Transactions on Electronic Computers, EC11 nº6, p.334-339, junho, 1962.

7- MARTINS, M. Ferreira - "Projeto de uma Unidade Aritmética para uma UCP de Médio Porte", Rio de Janeiro, NCE/UFRJ.

8- AUDE, J. Salek - "Projeto de um Sistema de E/S para uma UCP de Médio Porte", Rio de Janeiro, COPPE/UFRJ, 1978.

9- PACHECO, A. Cavalcanti Júnior - "Projeto de um Sistema de Memória "Cache - Backing" para uma UCP de Médio Porte", Rio de Janeiro, COPPE/UFRJ, 1979.

10- NCE - "Manual do Processador", Rio de Janeiro, NCE/UFRJ 1979.

11- NCE - "Manual de Manutenção do Processador", Rio de Janeiro, NCE/UFRJ, 1979.

12- PINTO, S. Brandão - "Testador de Circuitos Digitais", Rio de Janeiro, NCE/UFRJ, 1975.

13- VIANNA, R.A.S. Parente - "Projeto de um Painel, Carregador de Microprograma e Depurador Programável de Circuitos Digitais para uma UCP de Médio Porte", Rio de Janeiro, COPPE/UFRJ, 1978.

14- D.E.C. - "PDP 11/70 - Processor Handbook", Maynard, Massachussets, Digital Equipament Corporation, 1976.

15- D.E.C. - "PDP - Pheripherals Handbook", Maynard, Massachussets, Digital Equipament Corporation, 1976.

16- D.E.C. - "KB11-C-Processor Manual (PDP11/70)", Maynard, Massachussets, Digital Equipament Corporation, 1976.

17- HUSSON, S.S. - "Microprogramming : Principles and Practices", Englewood Cliffs, New Jersey, Prentice - Hall Inc., 1973.

18- SALISBURY, A.B. - "Microprogrammable Computer Architectures", New York, American Elsevier Publishing Company Inc., 1976.