PROJETO DE UM PAINEL, CARREGADOR DE MICROPROGRAMA E DEPURADOR PROGRAMÁVEL DE CIRCUITOS DIGITAIS PARA UM COMPUTADOR DE MÉDIO PORTE

Rogério Antonio Sampaio Parente Vianna

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:

Prof. Jayme Luiz Szwarcfiter
(Presidente)

Prof. Nelson Maculan Filho

Prof. Ivan da Costa Marques

RIO DE JANEIRO, RJ - BRASIL

DEZEMBRO DE 1978

AGRADECIMENTOS

Aos amigos do NCE/UFRJ pela paciência e dedicação.

RESUMO

A presente tese versa sôbre parte de uma UCP de porte médio compatível de software com o PDP-11/70. Tal projeto se encontra em desenvolvimento no NCE/UFRJ.

As partes aqui objetivadas são o Painel da máquina e dois sistemas de suporte do desenvolvimento e depuração da UCP, sediados no Terminal Inteligente do NCE: o Carregador de Microprogramas e o Depurador Programável de Circuitos Digitais.

ABSTRACT

This work presents the author's involvement on the conception of a medium seale mainframe, software compatible with Digital's PDP-11/70. The resulting prototype was developed at the NCE/UFRJ laboratories.

The thesis itself embraces studies on the panel processor and two auxiliary systems dedicated to the CPU's development and debbuging: the Micro-program Loader and the Digital Circuit's Programable Debugger, both running at the NCE's Intelligent Terminal.

Í N D I C E

PÁGINA

1 - INTRODUÇÃO GERAL 1

2 - APRESENTAÇÃO DO PROJETO 7

3 - PAINEL DA UCP 24

3.1 - Introdução 24
3.2 - Definição 26
3.3 - Descrição 35

4 - CARREGADOR DE MICROPROGRAMAS 42

4.1 - Introdução 42
4.2 - Definição do Sistema 45
4.3 - Interface "UNIBUS" / UNIDADE DE CONTROLE 52
4.4 - Definição do Software 56
4.5 - Conclusão 64

5 - DEPURADOR PROGRAMÁVEL DE CIRCUITOS DIGITAIS 67

5.1 - Introdução 67
5.2 - Definição do Sistema 72
5.3 - Descrição da Interface "TERMINAL INTELIGENTE/CIRCUITO EM DEPURAÇÃO 74
5.4 - Arquitetura do Software 77
5.5 - Conjunto de Instruções 82
5.6 - Exemplos de Programas de Depuração 92
5.7 - Conclusão 97

6 - BIBLIOGRAFIA 98

1 - INTRODUÇÃO GERAL

Este trabalho é parte de um projeto maior desenvolvido pelo NCE/UFRJ, qual seja o do projeto de um computador de médio porte.

Sabemos da necessidade de se produzir tais máquinas no país e dos esforços que vêm, desde já a algum tempo, sendo dispendidos para criar a tecnologia necessária. Diversos equipamentos já foram e continuam sendo desenvolvidos por várias equipes universitárias, num processo de criação não apenas do conhecimento técnico específico mas também do pessoal qualificado a manter esse esforço e difundi-lo tanto no meio profissional como no meio acadêmico.

Embora muito já tenha sido feito; o suficiente para garantir a posse dos conhecimentos básicos envolvidos e das medidas, ao nível de políticas para o setor, que se fazem necessárias à consolidação e aceleração tanto das pesquisas quanto da produção comercial dos equipamentos; quase tudo ainda está por fazer.

A inexistência de um parque industrial nacional, que absorva e multiplique a tecnologia já criada é certamente o fator de maior importância a ser considerado, na medida em que apenas ele pode "legitimar" as pesquisas,por estabelecer um ponto concreto de referência,mesmo para o trabalho acadêmico.

É esta a grande tarefa do momento: garantir a ocupação des

tes espaços com tecnologia brasileira.

Não é este no entanto o tema desta introdução. Estudos a esse respeito podem ser consultados através da bibliografia que apresentamos no final deste trabalho. Nos preocuparemos em mostrar que, embora nenhum (ou quase) produto brasileiro esteja sendo produzido regularmente, as "fronteiras" da pesquisa ainda podem ser alargadas, sem que se tornem irreais ou improdutivas, no sentido em que se estaria utilizando o escasso pessoal disponível em atividades que em nada estariam contribuindo para prover as técnicas necessárias à solução do problema básico que expusemos acima.

Embora essa questão esteja longe de ser simples, acreditamos que o desenvolvimento de uma máquina maior, mais complexa, se justifique basicamente através das seguintes considerações:

1) - É uma "pesquisa aplicada", ou seja, visa o desenvolvimento de um equipamento que tem a pretensão de vir a ser produzido comercialmente;

2) - É um equipamento que visa substituir similares estrangeiros, que atualmente satisfazem as necessidades do mercado;

3) - Seu processo de produção não envolveria técnicas não disponíveis no momento.

4) - No âmbito da pesquisa universitária é um "passo natural" à frente.

5) - Proporciona à equipe de trabalho, uma valiosa experiência dos problemas inerentes a um projeto de maior porte;

6) - Na medida em que é um trabalho desenvolvido por pessoal ligado à universidade, proporciona inestimável experiência a ser transmitida ao meio acadêmico.

Existem duas dúvidas ou objeções que se poderia levantar: uma, de que ainda existe uma considerável gama de equipamentos menores ainda não desenvolvidos ou consolidados.

Com efeito tal é o caso. No entanto pensamos que boa parte desses equipamentos não devem mesmo (exceto em circunstâncias especiais dependentes da instituição de pesquisa que o considerar) ser desenvolvidos nas Universidades. São, na grande maioria, equipamentos destinados a satisfazer mercados específicos e seu desenvolvimento na universidade seria descabido, na medida em que esta se transformaria em centro de pesquisa e desenvolvimento diretamente atrelada a mercados e fabricantes, que estariam simplesmente transferindo para uma instituição pública suas responsabilidades na área, obtendo os produtos a custo nulo, às expensas da sociedade. Nesta área, as diretrizes da pesquisa pública deve seguir uma política visando a criação tecnológica essencial ao rompimento das dependências existentes e não à satisfações mercadológicas, em grande parte criadas pelos próprios fabricantes. A questão é complexa e não nos alongaremos mais, existe, no entanto, ainda um outro ponto a considerar:

o projeto que ora apresentamos, longe de esgotar as necessidades existentes, é apenas um dos primeiros passos nessa direção e que, caso não seja seguido por outros projetos semelhantes, dificilmente ultrapassará a etapa da "expectativa geral", onde costumam dormitar os projetos, assim que deixam a "casa de seus pais".

A segunda objeção a que nos referimos é relativa à capacidade de se desenvolver a máquina.

Há realmente aqui uma objeção que não pode ser de todo resolvida. E isso se devem não a fatores genéricos, mas a uma limitação do NCE, que não possui no momento condições de desenvolver uma máquina desse porte e que seja original, ou seja, que necessite de criação tanto do hardware quanto do software necessários. Além disso, o presente projeto não conta, no momento, senão com a iniciativa do próprio NCE, não havendo a possibilidade de se realizar um trabalho conjunto com outros centros.

Uma decisão foi então tomada: a máquina não será original.

A opção de uma UCP software-compatível com alguma outra é discutível, mas tal é o caso: a UCP é software-compatível com o PDP-11/70 da DEC e visa substituí-lo inteiramente.

Naturalmente que não se trata de uma cópia, a arquitetura foi em parte redefinida, soluções novas foram encontradas para sua implementação e sua "fronteira" foi estendida com a

introdução de um Processador Periférico, além disso a unidade de processamento flutuante (PPF), que no PDP é opcional, foi incorporada à própria UCP.

Evidentemente não é essa a solução ideal, principalmente quando se considera o afastamento de todo o desenvolvimento de software. O que, acreditamos, valida o esforço é a necessidade de se produzir, o mais cedo possível, uma UCP de médio porte que "sugira" a reserva desses produtos à tecnologia nacional e que encorage outros empreendimentos nessa mesma área.

As questões que levantamos visaram apenas situar o problema, e de certo modo, justificar o projeto, estudos e debates sobre esses assuntos continuam sendo feitos e uma bibliografia dos mesmos é apresentada no final deste trabalho.

As características básicas do sistema são as seguintes:

- UCP de 16 bits;
- PPF em 32 ou 64 bits (precisão simples e estendida);
- Três (3) espaços de endereçamento "independentes": Usuário, Supervisor e Kernel, subdivididos em dois (2) subespacos: I (Instrução) e D (Dados);
- Controle microprogramado;
- Sistema de memória de até 4 MBytes do tipo "Cache-Backing";
- Barra padrão assíncrona de Entrada/Saída;
- Processador Periférico para o controle de periféricos lentos;

- Oito (8) modos de endereçamento;
- Aproximadamente 400 instruções.

Essas e outras características do projeto são apresentadas nas seções seguintes.

A apresentação do projeto, que se segue, visa dar uma visão de conjunto do mesmo, não é no entanto, possível descrevê-lo com detalhes, que poderão ser encontrados nos demais trabalhos de tese que o compõem.

2 - APRESENTAÇÃO DO PROJETO

Nessa apresentação pretendemos expor brevemente a arquitetura do sistema, a divisão do projeto entre a equipe e a descrição geral deste trabalho.

Não é possível oferecer-se neste trabalho uma descrição mais detalhada do PDP-11/70, que assumiremos ser devidamente conhecido. As consultas que ao mesmo se fizerem necessárias, podem ser realizadas nos próprios manuais da máquina que se encontram disponíveis no NCE/UFRJ.

A Figura 1 mostra a arquiterua geral do sistema. Podemos dividí-lo para efeito de apresentação em quatro (4) subsistemas:

1) - UCP;
2) - RELOCADORES;
3) - SISTEMA DE MEMÓRIA;
4) - PROCESSADOR PERIFÉRICO.

Como no sistema PDP-11/70 a UCP é uma máquina de 16 bits, que no nosso caso contém, ela própria, o Processador de Aritmética Flutuante (PPF).

O espaço de endereçamento é ampliado de 64 KBytes (16 bits) para 4 MBytes (22 bits) através de um sistema de relocação dos endereços virtuais.

O sistema de memória é do tipo "cache-backing", sendo o "cache" dimensionado para 8 KBytes. É implementado segundo

a técnica de mapeamento por "set-associativo", de associatividade igual a 2.

O Processador Periférico é o sistema (programável) que perfaz toda operação de E/S com periféricos lentos. Constitui-se de um microcomputador "Unibus-compatível" e de um sistema de interfaceamento com a barra de E/S da UCP e com o sistema de memória.

Como é usual, podemos dividir a Unidade Central de Processamento (UCP) em três (3) partes: uma Unidade Aritmética (UA), uma Unidade de Controle (UC) e outra de Entrada e Saída (UES).

Devido à inserção do PPF na própria máquina (em toda a linha PDP o PPF é opcional), as barras internas da Unidade Aritmética são de 64 bits (necessário para as operações flutuantes), utilizando as operações inteiras os 16 bits menos significativos destas barras.

Nesta arquitetura destacam-se os Registros Gerais em número de 16 e que são os seguintes:

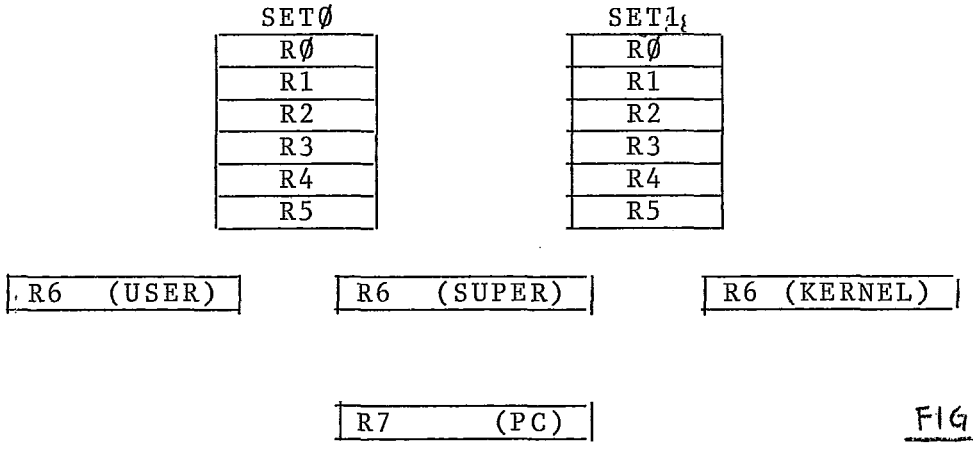


FIG.2

Esses registros são os únicos que podem ser referenciados implicitamente e que servem de base para o cálculo dos endereços dos operandos. Por exemplo a instrução ADD @(RØ)+ , @-(R3) realiza a soma de dois números cujos endereços são "calculados"da seguinte maneira:

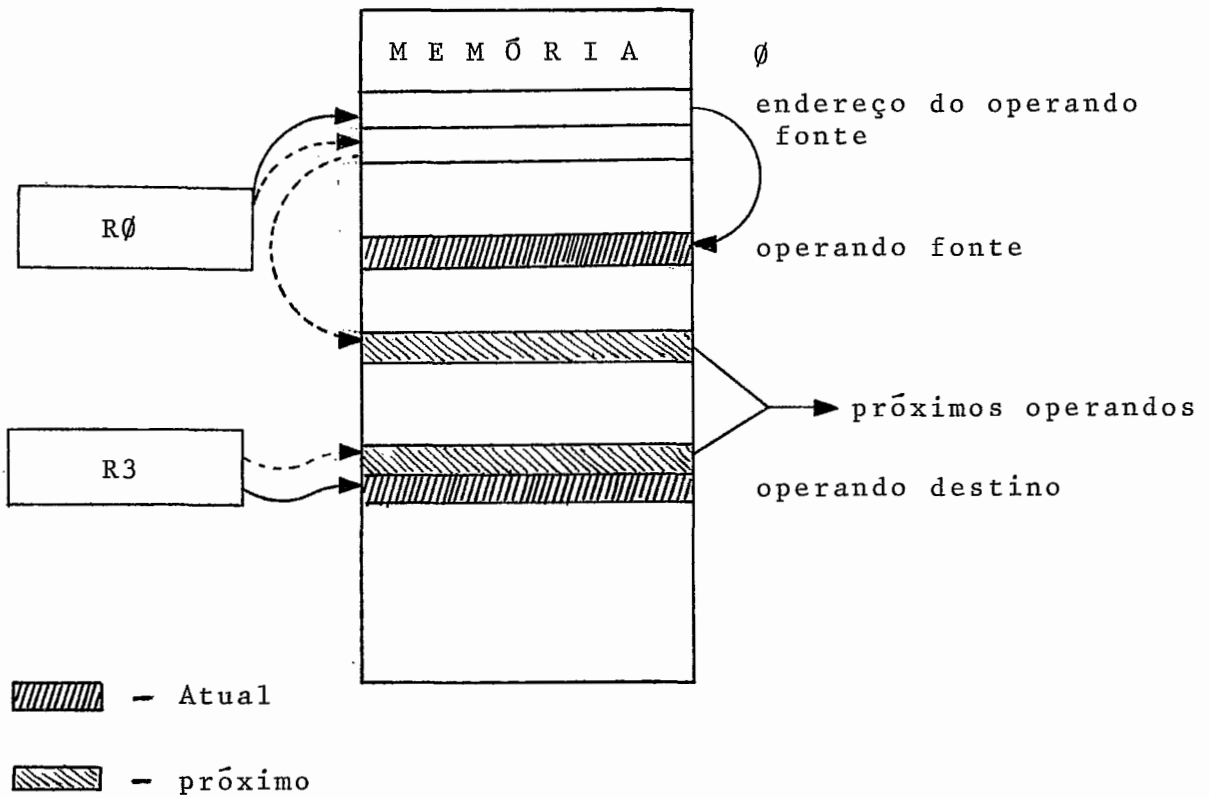


FIG. 3

No caso acima RØ será incrementado (de 2) no final da execução (pós-incrementado), R3 será decrementado (de 2) antes da busca do operando (pré-decrementado), e o resultado é armazenado no mesmo local onde se encontrava este último operando (@(R3)).

Uma descrição dos modos de endereçamento é apresentada mais adiante.

Os seis (6) primeiros registros estão duplicados em dois conjuntos ("sets"). O "set" é escolhido pelo sistema através da PSW ("Processor Status Word").

Os registros R6, embora possam ser usados como os demais, são interpretados como "Stack Pointers", havendo um para cada espaço de endereçamento.

Finalmente o registro R7 é único por ser o próprio Contador de Programa (PC). Embora hajam "restrições lógicas " quanto a seu uso, não existem "restrições físicas" que impeçam sua utilização normal.

Outro ponto a se destacar na UA é a presença das "macro-operações" utilizadas para perfazer rapidamente certas instruções inteiras e flutuantes.

São quatro (4) essas "macro-operações":

- equalização;
- normalização;
- multiplicação; e
- divisão.

São operações que, se microprogramadas, se tornariam muito lentas, e que então são implementadas à parte, através de um controle automático, funcionando na "escala de tempo do relógio principal". O microprograma apenas inicializa e "dis

para" essas "macros". Trata-se de uma "hibridização" do tipo de controle: convencional ("wired") para essas operações; microprogramado para as demais.

Como já adiantamos, o controle é centralizado numa Unidade de Controle microprogramada.

Usualmente, um Simulador da UCP é projetado com vistas ao projeto e "teste" dos microprogramas que depois são gravados numa memória de leitura exclusiva (ROM). No nosso caso, a Memória de Microprograma será inicialmente implementada com memórias de acesso aleatório (RAM), sendo os microprogramas depurados totalmente na própria máquina.

Surge então a necessidade de um sistema externo que permita a escrita e leitura dessa memória, e que ofereça facilidades a essa tarefa de depuração. Criou-se então um "Carregador de Microprograma", sistema que "roda" num Terminal Inteligente do NCE, apresentado na seção Nº 4, deste trabalho.

A comunicação da UCP com seus periféricos é, tanto no PDP como nesta máquina, efetuado pela própria UCP, não havendo processadores independentes de Entrada e Saída.

Os registros internos dos periféricos têm cada qual seu endereço próprio e a UCP os acessa por instruções usuais de transporte (MOV).

Esses endereços, reservados para periféricos, são os 8 KBytes mais significativos do espaço de endereçamento (3,992 MBytes

4 MBytes). Quando o resultado da relocação de um dado endereço virtual (gerado pela UCP) "cai" sobre essa faixa, a Unidade de E/S inicia um protocolo sobre a barra de E/S.

No nosso caso um Processador Periférico foi desenvolvido visando melhorar a performance do sistema pela transferência para esse processador, de todas as operações de E/S com periféricos lentos, que passarão a se comunicar apenas com esse processador.

Esse Processador Periférico (PP) é um microcomputador construído com uma pastilha microprocessadora: o INTEL-8080 , e de um interfaceamento com a Memória Principal do sistema e com a barra de E/S.

Diferentemente de um Canal Multiplex, esse processador pode executar tarefas mais sofisticadas (como operações condicionais, tratamento de erros, etc.) que permitem aumentar consideravelmente a performance do sistema.

Naturalmente sua utilização depende de substanciais modificações no Sistema Operacional e por esse motivo, sua implantação apenas se efetuará numa segunda fase, quando também será desenvolvido o software para o PP.

Os relocadores a que nos referimos são dois (2):

- o RELOCADOR DA UCP; e
- o RELOCADOR DA BARRA DE E/S.

Ambos se constituem de tabelas acessíveis para escrita e lei

tura pela UCP e que mapeiam, transformam, um endereço virtual (lógico) num endereço real (físico).

No primeiro caso, existem três (3) tabelas de relocação, uma para cada modo de endereçamento: Usuário (USER), Supervisor (SUPERVISOR) e Kernel (KERNEL):

Basicamente essa relocação é efetuada da seguinte maneira:

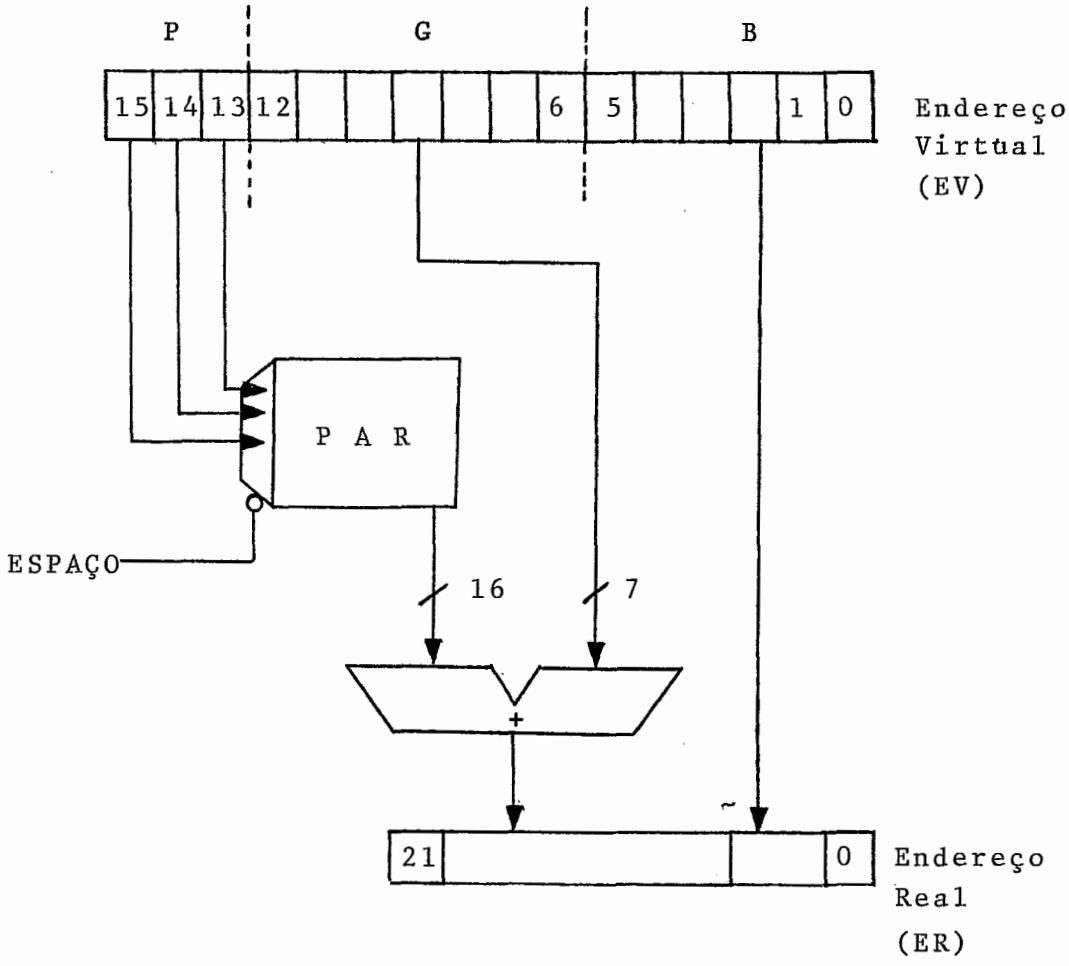


FIG. 4

Os três (3) bits mais significativos acessam um PAR ("Page Address Register") que "apontam" o endereço inicial desta página. Esses 16 bits são adicionados aos bits "G" (EV<12:6>) para "apontar" um dos 128 grupos (blocos) de 32 bytes cada, e o byte realmente acessado é determinado pela concatenação desses 16 bits resultantes com os 6 bits que indicam esse byte, formando um endereço real de 22 bits.

Como o sistema admite três (3) tipos de relocação: de 16 bits (sem relocação); 18 bits e 22 bits, uma série de modificações nesse esquema são necessárias, no entanto, a "filosofia geral" é mantida.

Na unidade existem três (3) conjuntos de PAR: um para cada espaço de relocação, e, além disso, cada espaço possui não 1 mas 2 conjuntos de PAR: um para o espaço "I" (INSTRUÇÃO) e um para o espaço "D" (DATA). A definição de espaço "I" e "D" pode ser vista com detalhes no trabalho sobre Entrada e Saída. Associado aos PAR's existem os PDR ("Page Descriptor Register") que, como o nome indica, descrevem o tipo da página (não residente, leitura exclusiva, não-usado, leitura e escrita, etc.) em questão.

No segundo caso, o relocador da barra de E/S é o mapeador, para a memória principal, dos endéreços que a tentam acessar através da barra de E/S.

As leituras e escritas de periféricos na memória (através de uma operação NPR) acessam então essa memória após sofrerem

uma relocação. Dos 128KW mais significativos do espaço total de 4 MBytes, os últimos 4 KW são reservados para os periféricos e os 124 KW restantes, para operações com a memória.

O esquema geral para a relocação em 22 bits é o seguinte:

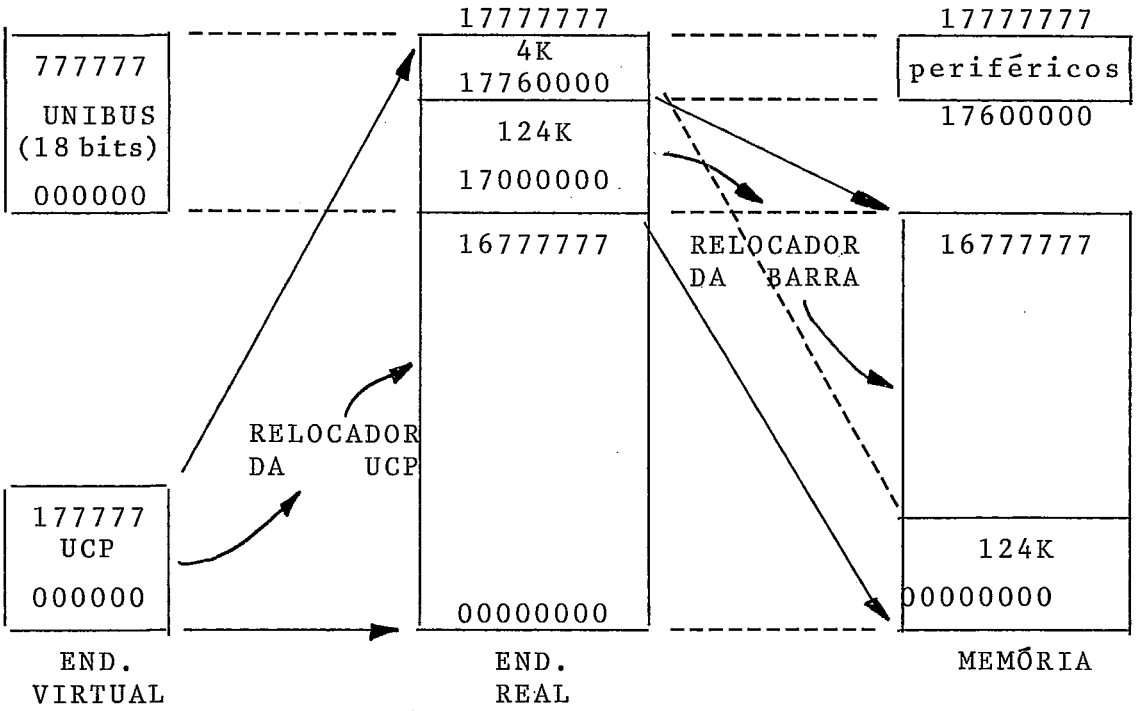


FIG. 5

Outro ponto a ressaltar é o esquema de interrupção. É constituído de 4 níveis "por hardware", ou seja, através da barra de E/S (BR = "Bus Request") e de 7 níveis internos (chamados de "interrupção por software") denominados PIR ("Program Interrupt Request"). Como de hábito, apenas no final da execução de cada instrução esses pedidos são anali-

sados e, de acordo com um esquema de prioridades (a própria UCP pode determinar seu nível de prioridade não aceitando pedidos de mais baixa ordem) um deles é selecionado para atendimento.

A barra de E/S a que nos referimos é um conjunto de 56 fios (chamados no PDP de "UNIBUS") que, de acordo com um protocolo determinado, provêm a comunicação da UCP com seus periféricos.

É uma barra assíncrona e sua descrição é a seguinte:

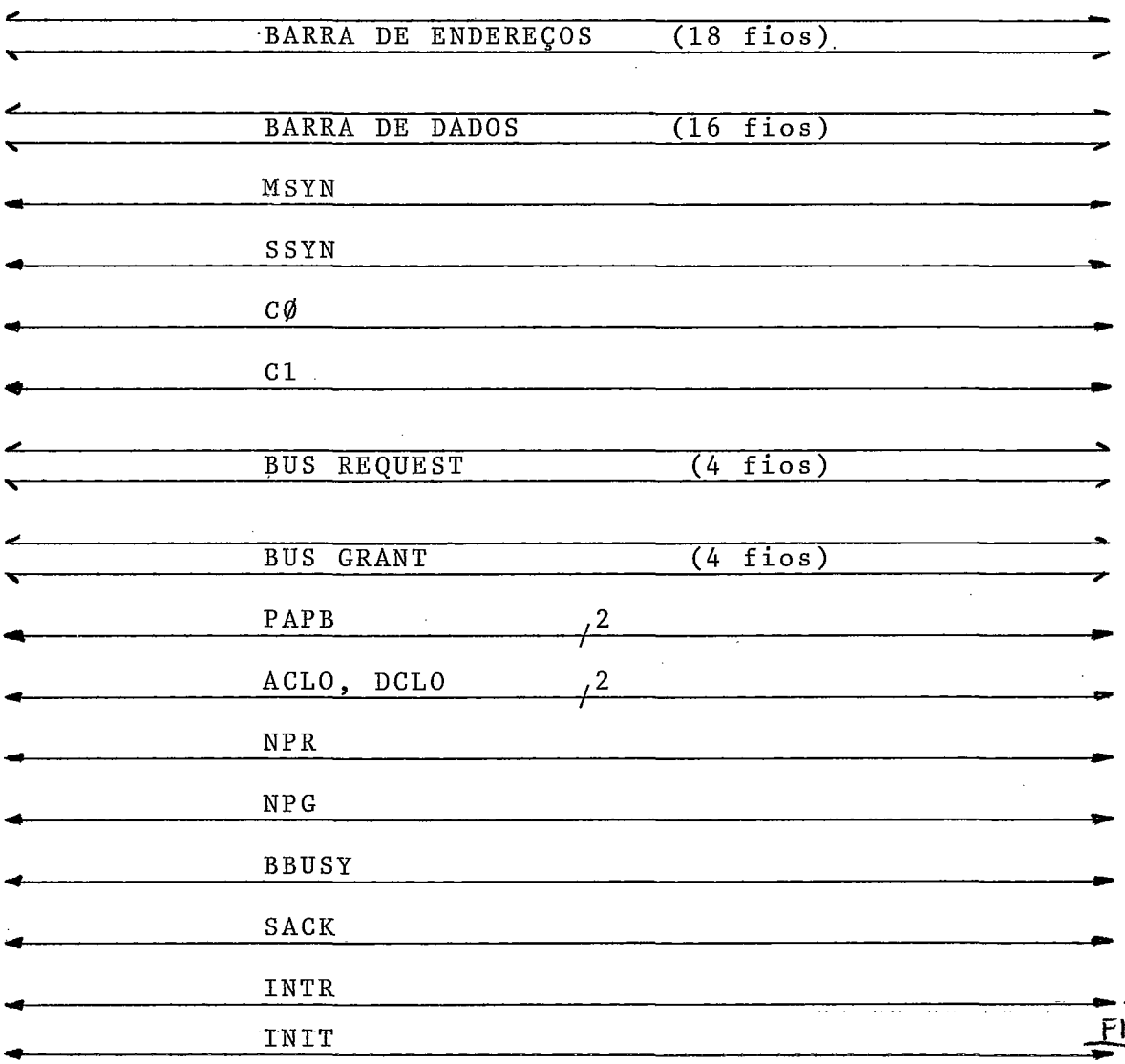


FIG. 6

1) - Barra de Endereços de 18 fios (18 bits —— 256KBytes);

2) - Barra de Dados de 16 Bits;

3) - MSYN - é o sinal que o periférico "Master", ou seja, o que comanda a transferência, envia para assinalar o envio (no caso de escrita - transferência para o periférico "slave") ou o recebimento (no caso de leitura), de um dado;

4) - SSYN - é o sinal enviado pelo periférico "slave" para indicar o recebimento do MSYN, e a finalização dessa transferência;

5,6) - CØ e C1 - indicam se a operação é de escrita ou leitura, de byte ou de palavra, ou de leitura sem reescrita (DATIP) (no caso da leitura em memória de núcleo magnético e que não se deseja que o sistema de memória reescreva o dado que é sempre "perdido" numa operação de leitura).

A codificação é a seguinte:

CØ	C1	
Ø	Ø	- DATIP
Ø	1	- DATOB
1	Ø	- DATI
1	1	- DATO

7) - BR - linhas de interrupção;

8) - BG - linhas de resposta aos pedidos de interrupção;

9) - PA, PB - linhas de erro de paridade;

10) - ACLO, DCLO - linhas indicativas de queda de tensão AC e DC;

11) - NPR - "Non-Processor-Request" - sinal utilizado para transferência de/ou para a memória, ou entre periféricos, operações que não necessitam interromper a UCP;

12) - NPG - "Non-Processor-Grant" - resposta do dispositivo acessado por um NPR;

13) - BBUSY - "Bus-Busy" - sinaliza que a barra de E/S está ocupada;

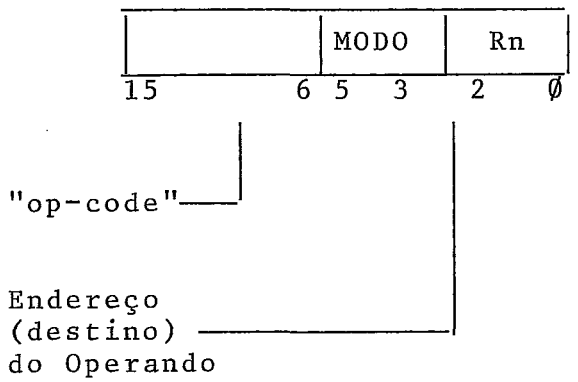
14) - SACK - "Slave-Acknowledge" - respota do "master" ao recebimento de NPG ou de BG;

15) - INTR - "Interrupt-Request" - sinaliza a existência de um pedido de interrupção;

16) - INIT - sinal de inicialização.

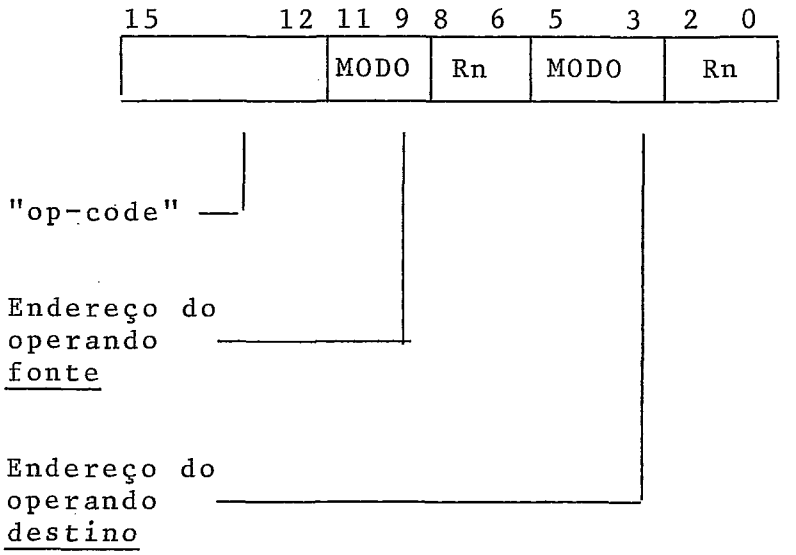
O formato das instruções são basicamente dois (2),referentes às instruções de um (1) e dois (2) operandos:

/ UM (1) OPERANDO /:



Exemplo: INC - ØØ52 nn (incrementa de 1 o operando).

/ DOIS (2) OPERANDOS /:



Exemplo: ADD - Ø6 mmnn; soma o operando fonte ao destino e guarda o resultado no endereço destino.

Resumidamente descritos, são os seguintes os oito (8) modos de endereçamento da UCP:

MODO Ø - Rn

Registro. O registro contém o operando.

MODO 1 - (Rn) ou @Rn

Indireto sobre o registro. O registro contém o endereço do operando.

MODO 2 - (Rn) +

Indireto pós-incrementado. O registro contém o endereço do operando, após a referência o registro é incrementado.

MODO 3 - @(Rn)+

Duplamente indireto pós-incrementado. O registro aponta o endereço do operando e após a referência é incrementado (de dois (2), mesmo em instruções de byte).

MODO 4 - -(Rn)

Indireto pré-decrementado. O registro é decrementado e então aponta o operando.

MODO 5 - @ - (Rn)

Duplamente indireto pré-decrementado. O registro é decrementado (sempre de 2) após o que, aponta o endereço do operando.

MODO 6 - X(Rn)

Indexado. O valor X (que segue a instrução) é adicionado à Rn para produzir o endereço do operando. Nem X nem Rn é modificado.

MODO 7 - @X(Rn)

Indexado e Indireto. O valor X é adicionado a Rn para apontar a posição de memória que contém o endereço do operando: X e Rn não sofrem modificações.

Dos modos de endereçamento, quando aplicados ao Registro R7, ou seja, o PC, são utilizados os quatro (4) abaixo:

MODO 2 - $(PC)_+$ - #n

Imediato. O operando segue a instrução.

MODO 3 - @(PC)$_+$ - @#A

Absoluto. O endereço do operando (absoluto) segue a instrução.

MODO 6 - X(PC) - A

Relativo. O operando se encontra X bytes distantes da instrução.

MODO 7 - @X(PC) - @A

Relativo Indireto. A uma "distância" de X bytes da instrução se encontra o endereço do operando

Observe-se que sobre o PC os modos: Ø, 1, 4 e 5 teriam o seguinte significado:

MODO Ø - PC; o operando é o próprio PC (!).

MODO 1 - @PC; o operando é a própria instrução.

MODO 4 - -PC; o operando é anterior em 1 ou 2 bytes ao PC. Se estabeleceria então um "loop", nessa instrução,(se fôr de palavra, se fôr de byte, ocasionará um erro de endereçamento).

MODO 5 - @-(PC); o PC seria decrementado de 2 e apontaria então o endereço do operando, também poderia entrar em "loop".

Por questões de unidade e coerência, o projeto foi dividido em seis (6) partes, cada uma delas entregue à responsabilidade de um dos membros da equipe e que viriam a ser apresentados com o trabalho de tese.

Após um período de trabalho unificado na definição da arquitetura e de outros pontos básicos, essas seis (6) partes foram estabelecidas como se segue:

1) - UNIDADE ARITMÉTICA, macrocircuitos e microprogramas das instruções flutuantes;

2) - UNIDADE DE CONTROLE e microprogramas;

3) - UNIDADE DE E/S (Controlador da barra de E/S, registros internos e relocadores);

4) - SISTEMA DE MEMÓRIA;

5) - PROCESSADOR PERIFÉRICO;

6) - PAINEL, CARREGADOR DE MICROPROGRAMA e DEPURADOR PROGRAMÁVEL DE CIRCUITOS DIGITAIS.

Cabem aqui duas (2) observações. Primeiro, a Unidade Aritmética não se constitui num trabalho de tese, tendo sido desenvolvida à parte. Segundo, o Depurador Programável de Circuitos Digitais foi incluído no projeto devido à necessidade de infraestrutura do laboratório. À última dessas partes é que se refere o presente trabalho.

O Painel (circuitos, microprogramas, e lay-out externo) é o meio de comunicação direta, manual, entre o "operador" e a "máquina". Sua descrição é apresentada na seção 3 do presente trabalho.

O Carregador de Microprograma é o sistema que, sediado num Terminal Inteligente, permite que se comande a Unidade de Controle durante a fase de depuração. Esse comando, basicamente escrita e leitura da Memória de Microprogramas e controle da execução, são descritos na seção 4.

Finalmente, o Depurador Programável é um sistema(hardware/software) também sediado no Terminal Inteligente que, através de um conjunto de instruções especialmente criado para esse fim, permite a depuração de um circuito digital através de um "programa de depuração". É um sistema "on-line" (operador-sistema-circuito em depuração) que visa facilitar o processo de depuração de um circuito digital. É descrito detalhadamente na seção 5.

Não pretendemos expor esses tópicos senão com os detalhes necessários à sua compreensão geral, os circuitos e programas específicos, assim como as descrições detalhadas dos mesmos podem ser encontradas nos manuais da máquina.

3 - PAINEL DA UCP

3.1 - Introdução

O Painel de uma máquina é um conjunto de chaves e lâmpadas que possibilitam o controle manual da UCP e que possui basicamente três (3) funções:

- a inicialização da máquina;
- auxílio à manutenção; e
- auxílio ao usuário na depuração (e manutenção) de programas.

Em máquinas de menor porte essa última característica é amplamente explorada principalmente naqueles casos em que o Sistema Operacional não é executado na própria máquina ("cross-software") ou mesmo, não existe um Sistema Operacional, casos em que o painel assume grande importância no auxílio à depuração de programas. Em máquinas maiores, no entanto, esse procedimento ou fica restrito à uma classe específica de usuários, os analistas que desenvolve software básico para a máquina (e mesmo assim com restrições), ou então é completamente abandonado, sendo o painel utilizado apenas nos dois primeiros casos citados.

É no auxílio à manutenção da máquina (principalmente em casos de pane) que o painel se mostra mais importante, e é devido a essa necessidade que sua complexidade tende a crescer.

Não iremos aqui discorrer genericamente sobre o Painel ou apresentar painéis de diversas máquinas. Estabeleceremos um conjunto de funções (basicamente as mesmas do PDP-11/70), as descreveremos e mostraremos, simplificadamente, como foram implementadas.

É preciso, no entanto, apresentar o que se pode considerar ser a principal característica do Painel, ou seja,que suas funções, exceto as que controlam diretamente o hardware (como veremos adiante), são microprogramadas. Isso implica que a manutenção não pode ser totalmente feita pelo painel , já que as funções de painel exigem, para sua execução, do bom funcionamento de parte considerável da máquina. Isso, no entanto, é natural, visto que a finalidade do painel não é prover capacidades tão acuradas. que, caso sejam necessárias, devem ser realizadas por outros equipamentos, particularmente, por circuitos especiais, desenvolvidos pelo próprio fabricante, e, que são acoplados à máquina, apenas naqueles casos em que se necessita de um outro "meio de controle" do computador e "independente" deste.

Para atender às necessidades de controle da execução de programas se introduz circuitos que comandam diretamente o tipo de execução da Unidade de Controle da UCP. Esses controles (que serão descritos adiante) são os usuais da técnica de execução sincronizada a eventos (comandos) externos e são principalmente quatro (4):

- execução de uma microinstrução por vez;

- execução entre operações de entrada e saída;
- execução até uma microinstrução determinada; e
- execução "clock a clock".

O Painel é ainda provido de indicadores (luzes) especiais que mostram sinais importantes ("status") da máquina, além de multiplexores que permitam a observação dos "caminhos de dados" e registros mais importantes, e de controle sobre os "modos de endereçamento" existentes (no caso do PDP-11/70 são três (3) esses modos: Kernel, User e Supervisor; subdivididos em dois (2) espaços: "I"(Instruction) e "D" (Data)).

3.2 - Definição do Painel

Dividiremos a definição em seis (6) partes:

1 - MODOS DE FUNCIONAMENTO DA UCP

A UCP pode ou não estar sob o controle do Painel, para isso se dispõe de uma chave que transfere esse controle. Às duas (2) posições dessa chave denominaremos (por ser a denominação usual) de HALT e ENABLE, a primeira correspondendo ao controle da UCP pelo Painel.

2 - FUNÇÕES MICROPROGRAMADAS

São cinco (5) essas funções:

a - EXAMINE (EXAM) - que coloca nos indicadores de dados o conteúdo do endereço previamente determinado;

b - DEPOSITE (DEP) - que coloca no endereço especificado um determinado dado, presente nas chaves de dados;

c - CARREGUE ENDEREÇO (END) - que especifica o endereço de memória ao qual se referirão as funções (a), (b) e (d);

d - PARTIDA (START) - que inicia a execução a partir do endereço especificado por (c);

e - CONTINUE (CONT) - que torna a executar o programa que havia sido interrompido pelo painel (ou por uma instrução de HALT).

3 - MODOS DE EXECUÇÃO

São três (3) os modos que podem ser escolhidos pelo Painel:

a - POR MICROINSTRUÇÃO (SINST);

b - POR CICLO DE ENTRADA E SAÍDA (SBCL) - Estas 2 (duas) possibilidades são exclusivas e sua escolha é feita por uma única chave no Painel;

c - POR CICLO DE RELÓGIO (SCLK) - pode ser usado em conjunto com as opções (a) ou (b) e um botão externo libera, cada vez que é acionado, um ciclo do circuito de "timing".

4 - MODOS DE VISUALIZAÇÃO

O Painel, através de luzes, mostra tanto endereços como dados (conteúdos).

Para a observação de endereços temos duas (2) possibilidades:

a - ENDEREÇO VIRTUAL - aquele gerado, pelo programa em execução, na Unidade Aritmética; e

b - ENDEREÇO REAL - aquele que, após ser relocado, atua realmente na memória (ou registros internos ou posições na barra de entrada e saída).

Para a observação de dados existem quatro (4) escolhas:

a - REGISTRO DE COMUNICAÇÃO (BR) - por onde passam todos os dados (códigos e operandos) que vêm do sistema de memória ou a ele se destinam;

b - REGISTRO DE LUZES (LR) - onde, por programa ou painel, é possível escrever-se um determinado valor;

c - REGISTRO DE ENDEREÇOS DE MICROPROGRAMA(ENDMP)- que indica a microinstrução atual.

d - REGISTRO DOS "OPERANDOS DESTINO" - por esse registro passam todos os valores finais de qualquer instrução, inclusive as instrução de modo destino igual a ZERO(Ø), ou seja, cujo valor final é armazenado num dos registros ge-

rais, não sendo transmitido para o sistema de memória.

5 - VISUALIZAÇÃO DE VALORES ESPECIAIS ("STATUS")

Para uma "completa" monitoração da máquina é necessário que se possa observar constantemente certos pontos dos circuitos, no nosso caso esses pontos podem ser colocados em três (3) grupos:

a - INDICADORES DE ERRO

- ERRO DE ENDEREÇAMENTO (EE); e
- ERRO DE PARIDADE (EP).

b - INDICADORES DE ESTADO DA UCP

- UCP EM EXECUÇÃO (EXEC); e
- UCP EM ESTADO DE ESPERA (PAUSA) - indica que a próxima operação não pode ser efetuada por falta de recurso (o recurso está sob controle externo, ou uma operação iniciada não obteve ainda a resposta esperada devido a atrazos nos circuitos).

- UCP CONTROLA A BARRA DE ENTRADA E SAÍDA (MASTER);

- UCP EXECUTANDO NO MODO KERNEL (KERNEL);

- UCP EXECUTANDO NO MODO SUPERVISOR (SUPER);

- UCP EXECUTANDO NO MODO USUÁRIO (USER);

- UCP EXECUTANDO SOBRE O ESPAÇO "D" (DATA) ou "I" (INSTRUÇÃO).

c - INDICADORES DE MAPEAMENTO:

- SISTEMA OPERANDO SEM RELOCAÇÃO (16 bits) (16);
- SISTEMA OPERANDO COM RELOCAÇÃO (18 bits) (18);
- SISTEMA OPERANDO COM RELOCAÇÃO (22 bits) (22).

6 - SELEÇÃO DO "ESPAÇO DE RELOCAÇÃO"

Como já vimos, existem três (3) espaços de endereçamento em que um programa pode ser executado, cada um deles subdividido em dois (2) outros: os espaços de Instrução(I) e Dados(D). É possível pelo Painel, selecionar qualquer um desses "modos de relocação" e, além disso, é ainda possível ao Painel, operar diretamente com endereços reais de 22 bits, caso em que se inibe o sistema de relocação. É ainda importante que o endereço real gerado pelo sistema de relocação a partir de um determinado endereço virtual seja observável, essa facilidade foi então incorporada ao Painel. Podemos então resumir:

a - ESPAÇO ATUAL DE RELOCAÇÃO = KERNEL I (KI);
b - ESPAÇO ATUAL DE RELOCAÇÃO = KERNEL D (KD);
c - ESPAÇO ATUAL DE RELOCAÇÃO = SUPER I (SI);
d - ESPAÇO ATUAL DE RELOCAÇÃO = SUPER D (SD);
e - ESPAÇO ATUAL DE RELOCAÇÃO = USER I (UI);
f - ESPAÇO ATUAL DE RELOCAÇÃO = USER D (UD);
g - SEM RELOCAÇÃO, ENDEREÇO REAL = CONSPHY (CP).

Essas possibilidades são selecionadas por uma chave externa de sete (7) posições (RS).

h - RELOCAÇÃO DETERMINADA PELA CHAVE ACIMA, MAS COLOCANDO NAS LUZES DE ENDEREÇO O ENDEREÇO REAL (PROGPHY).

São essas as funções que o Painel pode comandar mas existem ainda outras duas chaves, uma de POWER-ON-LOOK, que liga ou desliga a máquina além de, numa terceira posição inibir os controles do Painel para proteção contra usos indevidos, e outra de teste das lâmpadas do Painel.

Pode-se visualizar esses controles no esquema à seguir.

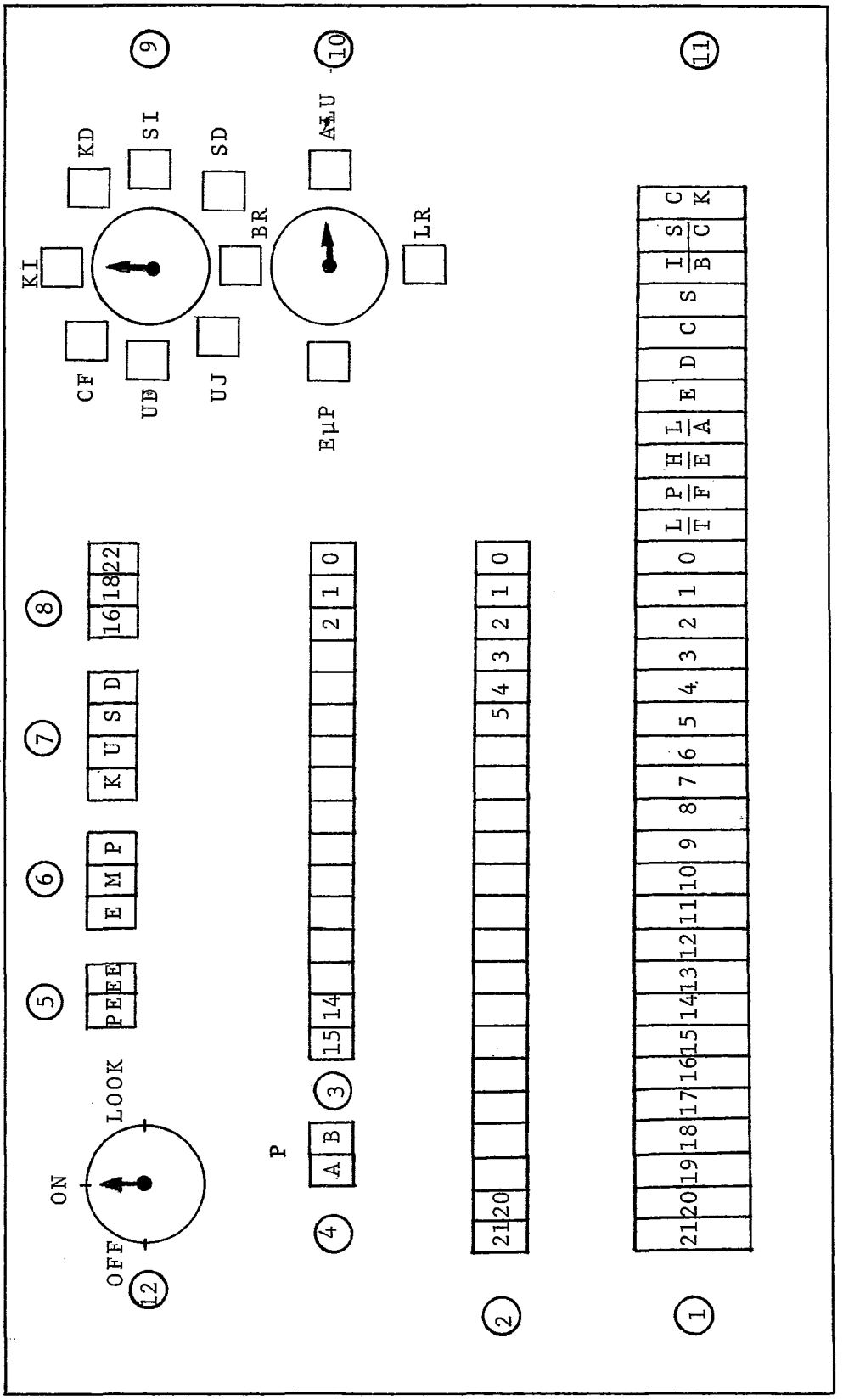

KD
SI
SD
ALU
KI
BR
LR
CF
UJ
EμP
LOOK
ON
OFF
P
E M P
K U S D
16 18 22
15 14
2 1 0
A B
21 20
5 4 3 2 1 0
21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
L/T
P/F
H/E
L/A
I/B
S/C
C K
9
10
11
12
8
7
6
5
4
3
2
1

FIG. 1

Descrição:

1 - Registro de Chaves (CH<15:00> + CH<21:16>);

2 - Luzes de Endereços;

3 - Luzes de Dados;

4 - Indicadores de Paridade (por bytes: A - Byte Alto; B - Byte Baixo);

5 - Indicadores de erro de paridade e endereçamento;

6 - "Status" da UCP: MASTER, PAUSA e EXECUTANDO;

7 - Espaço atual de relocação: KERNEL, USUÁRIO, SUPERVISOR, DADO (ou INSTRUÇÃO);

8 - Tipo de Relocação: 16 (sem relocação), 18 e 22 bits;

9 - Seletores de espaço de relocação: K, S, U; "I" ou "D" e "CONSOLE FISICO" (Endereço real direto das chaves(1));

10 - Seletores de Dado: A<15:00> (registro dos "Operandos destino" da UCO); LR:registro de luzes acessável por programa; EMP:microinstrução atual (endereço da memória de controle em 1Ø bits); BR:registro de comunicação;

11 - Chaves de comando, da esquerda para a direita:

LT - Teste das lâmpadas do Painel;

PF - Programa Físico: nas luzes de endereço o endereço real;

HE - HALT/ENABLE - Controle pelo Painel;

LA - Carregue Endereço (END);

E - Examine (EXAM);

D - Deposite (DEP);

C - Continue (CONT);

S - Partida (START);

IB - "Single Instruction" ou "Single bus cicle";

SC - "Single Clock";

CK - Tecla de repetição da função anterior.

12 - Chave LIGA-DESLIGA-LOOK.

3.3 - Descrição

Uma característica do Painel é a pequena relação funcional entre suas diversas partes, o que torna difícil uma esquematização mais geral.

O projeto pode ser dividido em cinco (5) partes:

- "Lay-Out" do painel externo;
- Cablagem;
- Circuitos;
- Microprogramas; e
- Descrição das funções e da operação.

A descrição do lay-out, da cablagem, dos circuitos e dos microprogramas exigem um nível de detalhamento na apresentação da máquina que foge ao escôpo deste trabalho. Optamos então, nessa descrição, pela exposição de alguns detalhes das funções do Painel que julgamos de interesse ressaltar.

Como vimos são cinco (5) as funções microprogramadas : Carregue Endereço, Leitura (Examine), Escrita (Deposite), Continuação e "Start". Na verdade existem seis (6) outras funções microprogramadas, "transparentes" ao operador. Duas delas se referem a operações repetitivas das funções de Leitura e Escrita, ou seja, é possível a leitura, ou escrita, de posições sucessivas, sem necessidade do uso da função Carregue Endereço a cada passo. É evidente que essa realização exige um microprograma diferente uma vez que é necessário o

incremento automático do endereço antes da operação de entrada e saída correspondente.

As outras quatro (4) funções se justificam por um detalhe de implementação dos Registros Gerais da UCP.

O acesso a esses registros é feito unicamente por instruções que os referenciam implicitamente, ou seja, todas as operações sobre esses registros se fazem sob o controle do microprograma. Não há pois, apesar dos registros serem considerados como pertencentes ao espaço de endereçamento da máquina (mais precisamente no espaço reservado à barra de entrada e saída), acesso aos mesmos através de uma operação de entrada e saída (um tal acesso geraria um "time-out").

Devido a essa particularidade,a escrita e a leitura desses registros devem ser feitas separadamente, ou seja, é preciso tornar "implícitos" esses acessos. Tal se realiza através de um decodificador do endereço proposto em Carregue Endereço, que identifique um acesso aos registros e os faça executar por um microprograma específico. A facilidade de repetição é também concedida, donde as quatro (4) funções adicionais.

Outra diferença importante entre esses registros e as demais "posições de memória" é que esses registros podem ter endereço impar. Isso se dá porque o espaçamento entre eles não é, como nos demais registros e memória de dois (2) bytes, mas apenas de um (1) byte. Um "acesso por byte" a es-

ses registros por definição atua apenas nos oito (8) bits menos significativos. Os microprogramas de operações repetidas sobre esses registros incrementam o endereço de apenas uma unidade a cada passo.

É preciso ainda observar que o acesso a esses registros pelo Painel, embora seja efetuado pelo operador de maneira igual aos demais acessos, não obedecem, por motivos que decorrem do exposto, às regras gerais, ou seja, para acessá-los é necessário que a função inicial Carregue Endereço "aponte" para os mesmos, o que significa que eles não são considerados contíguos (fisicamente) a quaisquer outros endereços, (sob esse ponto de vista). Decorre desse fato que as funções de repetição (escrita ou leitura) não os podem alcançar, além do que as funções de repetição executadas sobre esses registros não podem abandonar a área desses registros. Um exemplo simples (símbólico)esclarecerá esse fato: suponhamos a seguinte configuração dos conteúdos dos endereços abaixo:

ENDEREÇO	CONTEÚDO (Decimal)	
77767Ø	Ø	
777672	1	
777674	2	
777676	3	
7777ØØ (RØ, STØ)	4	
⋮	⋮	ÁREA DOS REGISTROS
⋮	⋮	GERAIS (SETØ, 1)

ENDEREÇO	CONTEÚDO (Decimal)
777717 (R6, USER)	19
77772Ø	2Ø
777722	21
777724	22
777726	23

As duas (2) sequências de operações abaixo esclarecem o problema:

	FUNÇÃO	CHAVES	LUZES DE DADOS	LUZES DE ENDEREÇO (real)
1 -	CAR. END.	777672	X	777672
	EXAMINE	X	Ø	777672
	EXAMINE	X	1	777674
	DEPOSITE	3Ø	3Ø	777674
	EXAMINE	X	3Ø	777674
	DEPOSITE	3Ø	3Ø	777674
	DEPOSITE	31	31	777676
	DEPOSITE	32	X	7777ØØ (TIMEOUT)
	EXAMINE	X	X	7777ØØ (TIMEOUT)
2 -	CAR. END.	777716	X	777716
	EXAMINE	X	18	777716
	EXAMINE	X	19	777717
	DEPOSITE	4Ø	4Ø	777717
	EXAMINE	X	4Ø	777717
	DEPOSITE	41	41	7777ØØ (LOOP)
	DEPOSITE	42	42	7777Ø1
	DEPOSITE	43	43	7777Ø2
	EXAMINE	X	43	7777Ø2

A implementação desses circuitos se faz segundo o esquema geral abaixo:

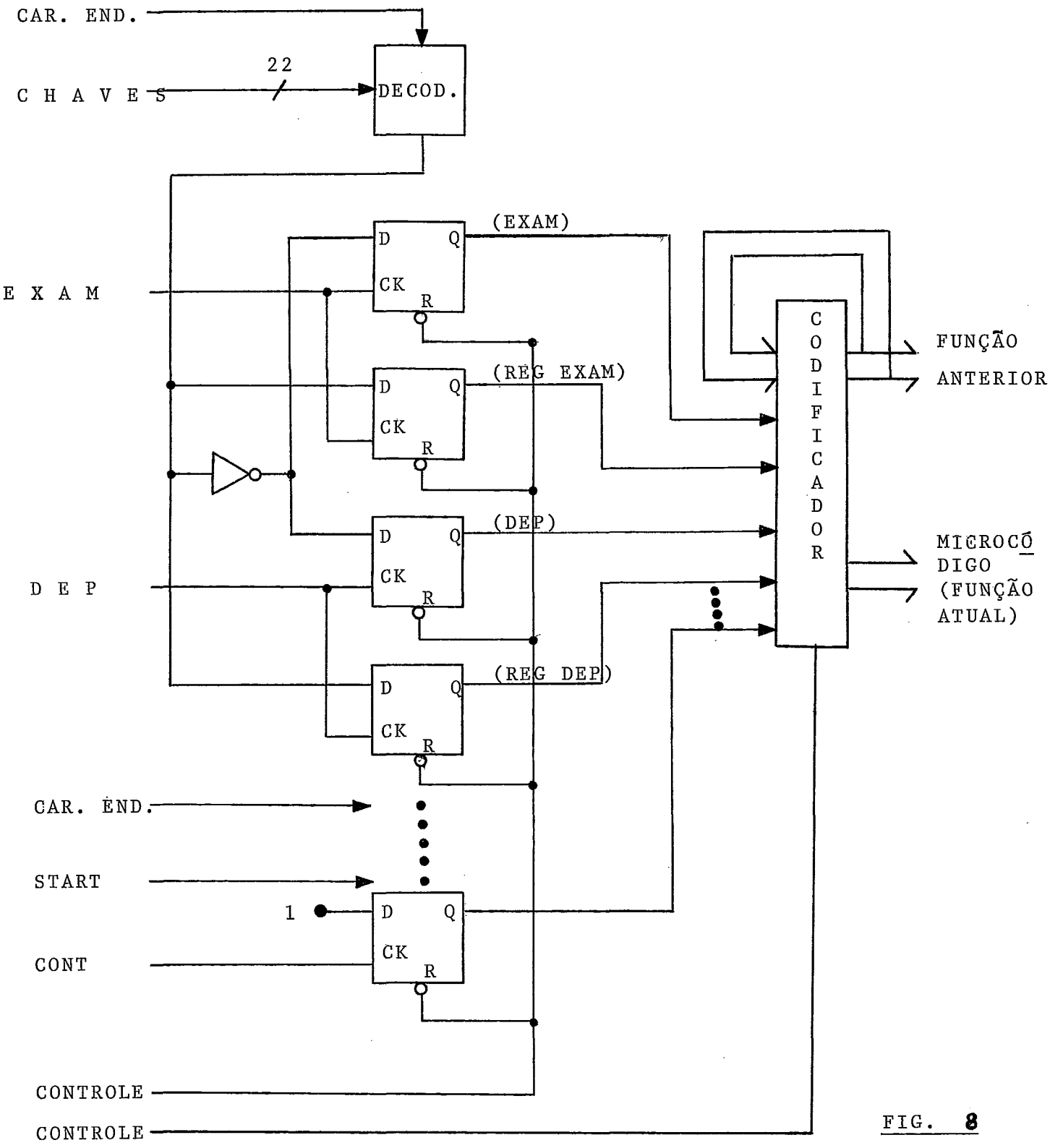


FIG. 8

A realimentação no circuito codificador determina as operações repetidas ("step").

Os microcódigos são os sinais que especificam a função a ser executada, excitam a "lógica de desvio" da U/C compondo o endereço da microinstrução a ser executada.

Os microprogramas das funções do Painel obedecem à seguinte organização geral:

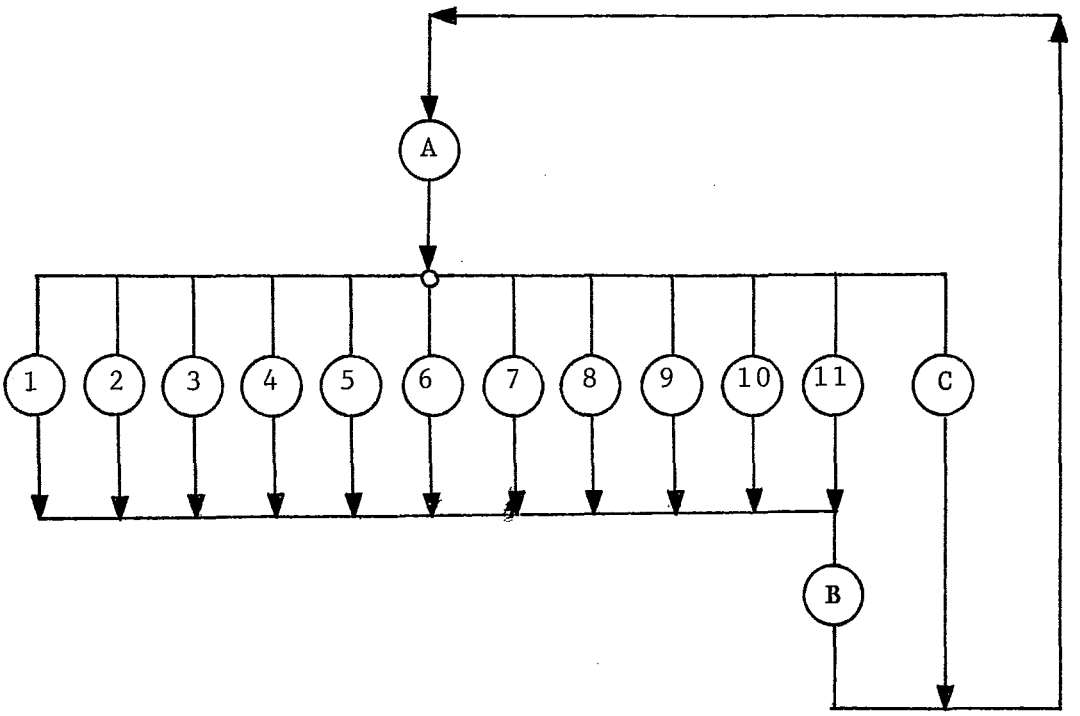


FIG. 9

Onde cada bloco assinalado corresponde ao microprograma de uma função específica:

1 - "START";

2 - "CONT";

3 - CAR.END.;

4 - DEPOSITE;

5 - "STEP"-DEPOSITE (2 ou mais operações);

6 - EXAMINE

7 - "STEP"-EXAMINE (2 ou mais operações);

8 - REG. DEPOSITE (escreve num registro geral);

9 - "STEP"-REG. DEPOSITE (2 ou mais operações);

1Ø - REG. EXAMINE (mostra nas luzes (DATA) o conteúdo do registro geral acessado);

11 - "STEP"-REG. EXAMINE (2 ou mais operações);

A - Trecho inicial comum (mostra o registro de chaves, etc.);

B - Trecho final comum (posiciona os circuitos para nova operação);

C - "No-function" (Aguarda uma função num "loop" de espera).

A descrição dos microprogramas envolve o conhecimento de todo o hardware necessário e não é nosso objetivo descer a tais detalhes, acrescentamos apenas que o esquema apresentado acima na verdade sofre a influência das características específicas dos microprogramas e que, devido a possibilidade de minimizações se modifica um pouco.

4 - CARREGADOR DE MICROPROGRAMA

4.1 - Introdução

Como já dissemos a memória de controle da UCP será inicialmente implementada com RAMs e os microprogramas serão depurados na própria máquina.

Normalmente essa depuração é parcialmente efetuada num simulador, principalmente nas primeiras etapas quando os microprogramas são projetados sendo depois testados na máquina.

Note-se que mesmo com o auxílio de um simulador o problema da depuração dos microprogramas subsiste, uma vez que o método, principalmente em máquinas relativamente grandes, não permite alcançar o "microprograma final" sem a necessidade de testes reais. Isso se dá basicamente porque na simulação dificilmente se consegue reproduzir todos os aspectos da máquina, principalmente no tocante às sincronizações ao nível do hardware e aos tempos envolvidos. Um simulador usualmente apenas fornece o valor dos registros e da memória a cada operação, de modo que se possa verificar a "correção lógica" dos procedimentos, sem no entanto se ocupar das sequências de operação ao nível do hardware que realmente ocorrem na máquina. Por esses motivos quase nunca é possível projetar e testar os microprogramas exclusivamente através de um simulador.

Muito embora fosse muito conveniente, não foi possível à equipe projetar um simulador. Por essa razão todo o proces-

so de testes será feito na própria máquina.

Para o protótipo final será feita a substituição de RAMs por PROMs (1) eliminando-se os inconvenientes que uma memória de controle volátil pode ocasionar.

Naturalmente essa substituição não é imediata. Várias modificações serão necessárias na fase de substituição (Vide ADRIANO JOAQUIM DE OLIVEIRA CRUZ - "Projeto de uma UCP de Médio Porte - Unidade de Controle" - Tese/COPPE/77), mas as condições que garantem o funcionamento da máquina final ficam asseguradas, visto que as diferenças existentes na execução não são relevantes.

Surge então o problema de se definir como será feita a carga do microprograma.

O "sistema de carga" deve ser prático, de modo a não criar embaraços e dificuldades ao pessoal da depuração, flexível, de maneira a permitir um controle total da unidade de controle e do próprio processo de depuração, e rápido, de modo que as modificações que se fizerem no microprograma não provoquem perda de tempo.

Um sistema programável é aquele que atende a essas necessidades.

Como realizar o acoplamento entre a unidade de controle e esse sistema é um outro problema que se apresenta.

A solução adotada foi utilizar um Terminal Inteligente

(2) para sediar o sistema e acoplá-lo à unidade de controle através do "unibus"; barra de Entrada e Saída da UCP.

O hardware adicional necessário ao sistema de carregamento se compõe de um decodificador de "unibus" e um controlador do processo de carga, ambos serão retirados do protótipo final.

Restava ainda definir a interação entre o sistema e o operador. Seria possível realizá-la através do teclado e do vídeo do TI, no entanto, optamos pelo acoplamento através de uma teletipo, o que forneceria ao operador um documento escrito do seu procedimento, facilitando análises e eliminando repetições desnecessárias, além de fornecer material para discurssões posteriores e deixar um "rastro" de todo o processo, o que é particularmente interessante num trabalho pioneiro, quando apenas no final, ou mesmo após, se possui condições de reavaliar todo o projeto e os métodos e técnicas utilizadas. A FIG. 1 mostra o esquema de acoplamento das várias partes do sistema.

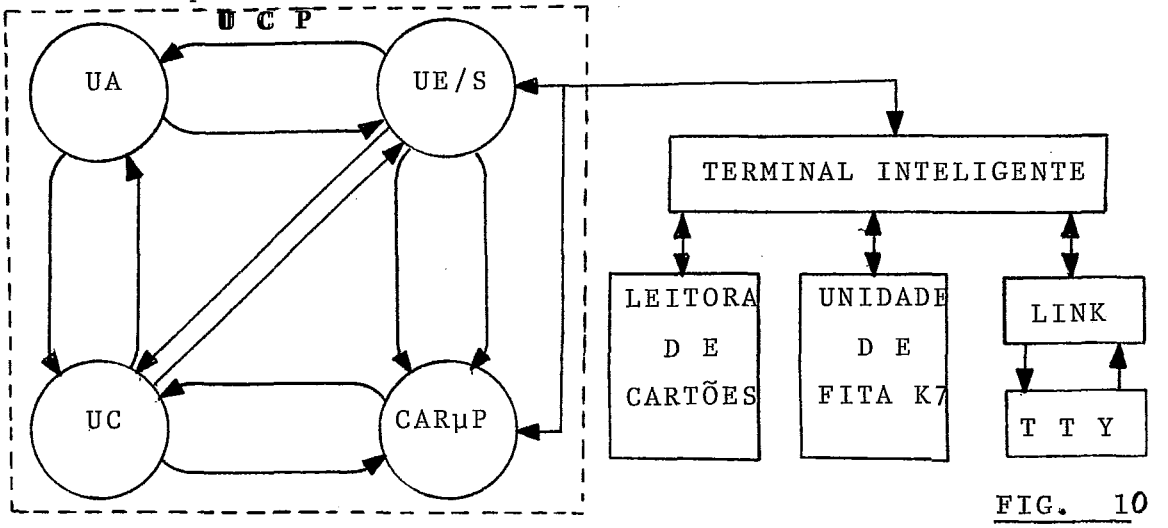


FIG. 10

4.2 - Definição do Sistema

O objetivo do sistema aqui apresentado é oferecer ao pessoal da depuração dos microprogramas e unidade de controle um método e um dispositivo capaz de permitir que essa depuração se faça, em todos os níveis, na própria máquina. Para isso é necessário carregar a memória de controle com os padrões de bits desejados das diversas maneiras necessárias a um procedimento simples e rápido, além de possibilitar o controle da execução desses microprogramas.

Definimos então 14 funções básicas para o sistema , que passaremos a descrever.

Antes porém, uma observação: a memória de controle da UCP foi implementada com 64 pastilhas de 1024 palavras de 1 bit. Uma microinstrução ocupa 2 palavras, sendo então essa memória de controle vista pelas demais unidades como uma organização de 512 palavras de 128 bits. Para o carregador, no entanto, essa memória é vista como sendo de 8 K palavras de 8 bits (3).

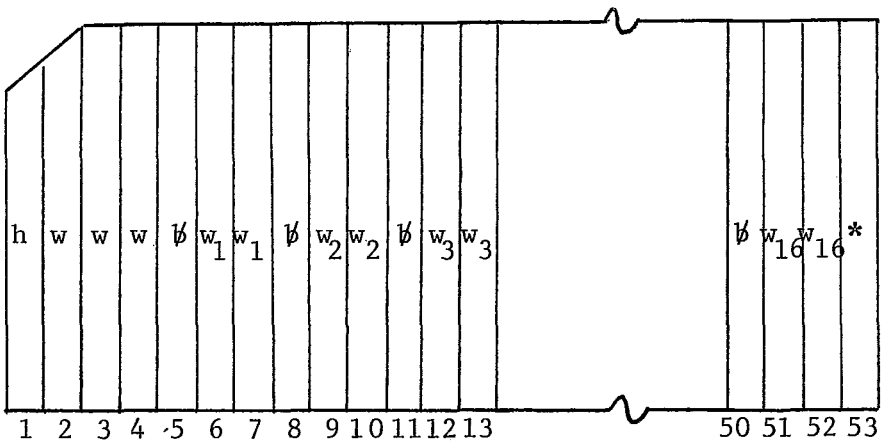
FUNÇÕES DO SISTEMA "CARREGADOR DE MICROPROGRAMA"

1) - Transferência para a memória de controle dos microprogramas contidos em cartão perfurado.

Nas etapas mais ulteriores da depuração haverá a necessidade de se transferir um grande volume de bytes, o que justifica a função, além disso é conveniente possuir cópias do microprograma num "arquivo" seguro.

O cartão foi então formatado de maneira a simplificar sua perfuração e entendimento. Como a microinstrução da UCP é de 128 bits, é natural então usar esse número, 16 bytes, para preencher um cartão.

Naturalmente, os bytes deverão ser perfurados em notação hexadecimal. Um cartão típico seria então:



$\overline{h}$ = Ø, 1

$\overline{w}$ = Ø, ..., F

FIG. 2

Onde os primeiros 4 caracteres especificam o endereço da memória de controle para onde será transferido o primeiro byte do cartão, estando os demais em sequência. Os brancos separam os bytes e o cartão é terminado com "*". Embora seja essa a formatação aconselhada, o cartão pode conter mais ou menos uma microins

trução, ou seja, mais ou menos que 16 bytes.

O cartão é então lido, decodificado e transmitido à memória de controle, byte a byte, até que a rotina de transferência acuse o "*" final e comande o próximo cartão.

Um "*" na primeira coluna indica o fim de todos os cartões.

2) - Transferência para a memória de controle do micropro - grama especificado pela TTY

É através da TTY que se especifica porções relativamente pequenas do microprograma e se faz as alterações, em caso de erro, nesse mesmo microprograma.

Nessa função a formatação é livre e o operador pode transferir quantos bytes desejar, que, como na função anterior, estarão em sequência e o primeiro deles transferido para o endereço da memória de controle especificado.

Ao final da especificação a "string" é então decodificada e transmitida para a memória de controle

3) - Leitura de uma microinstrução da memória de controle e sua visualização na TTY.

Para uma completa monitoração da memória de controle é indispensável observá-la e assim verificar as pastilhas danificadas, erros de fiação, mudanças de conteúdo devido à ruídos, etc. O operador fornece então o endereço base em 13 bits (4) (notação hexadecimal) e o sistema lê da memória de controle o byte acessado e os 15 seguintes e os envia à TTY.

4) - Transferir toda a memória de controle para uma unidade de fita K7 do TI.

5) - Transferir para a memória de controle o microprograma armazenado nessa fita K7.

Esses procedimentos se devem basicamente a três (3) considerações:

1) - A necessidade de se evitar descontinuidades no trabalho, assim se torna possível que as interrupções (diárias, por exemplo) não exijam um processo lento e penoso de reinicialização. Quando da suspensão, o microprograma (com todas as alterações efetuadas) é transferido para a fita e no momento de se reiniciar o trabalho efetua-se a transmissão oposta.

2) - A necessidade de se manter uma cópia do "microprograma atual". Estando essa cópia em fita K7 o operador se vê livre de manter em papel as alterações efetuadas, o que certamente ocasionaria grande perda de tempo e trabalho adicional.

3) - A conveniência de se possuir um meio prático de listagem do microprograma quando este estiver terminado. Assim, na documentação final, será possível obter essa listagem, no formato mais conveniente, simplesmente através de uma rotina que transferirá o microprograma da fita K7 para a TTY.

6) - Comandar a execução do microprograma

Após a carga da memória de controle com os microprogramas que se desejar depurar, é necessário liberar a execução da unidade de controle a partir de um determinado endereço daquela memória, que é então especificado pelo operador.

7) - Suspensão da execução do microprograma

O processo exigirá o exame, com os equipamentos usuais, dos circuitos da máquina, quer as unidades estejam ativadas ou não. Além disso, estabelecemos no protocolo, que as funções de transferência de dados somente serão

executadas se o operador tiver comandado a suspensão da execução, ou seja, se a máquina estiver parada.

Modos de Execução:

8) - Execução interrompida automaticamente após a execução da microinstrução cujo endereço for igual em valor ao conteúdo do "Registro de Comparação" (RC).

O "registro de comparação" é um registro de 9 bits (5), utilizado para, por comparação, comandar a suspensão da execução assim que a microinstrução respectiva houver sido executada. Pode-se, desta mameira, introduzir "pontos de quebra" na execução do microprograma, procedimento indispensável a um completo controle do processo.

9) - Executar apenas uma microinstrução.

10) - Executar até o início de um "ciclo de bus".

11) - Executar assincronamente.

12) - Continuar a execução (interrompida quer automaticamente, quer por comando externo).

Essas funções completam o esquema de controle total sobre a execução do microprograma.

Os três (3) primeiros modos de execução são os métodos

mais importantes de se introduzir "pontos de quebra" na execução, técnica indispensável à depuração de qualquer sistema.

Apenas a função número 10 necessita de uma explicação adicional: por "ciclo de bus" se entende, nessa máquina, à toda operação de transferência de dados entre a UCP e a memória ou barra de E/S. O controle deteta então uma operação desse tipo e paraliza a execução, permitindo a verificação das condições iniciais da operação (6).

13) - Listagem, na TTY, do microprograma armazenado na fita K7.

Essa função somente será implementada no final da depuração, quando os microprogramas estiverem totalmente prontos.

14) - Verificação de pastilhas danificadas

É realizada através de escrita e leitura de bytes significativos (/55 e /AA). O sistema sinaliza a ocorrência do fato e especifica a(s) pastilha(s) em questão.

4.3 - Interface UNIBUS/UNIDADE DE CONTROLE

É através dessa interface que o sistema carregador de microprograma comanda a Unidade de Controle da UCP.

Por estar sujeita ao protocolo de "unibus" e a ele conectada essa interface é um periférico da Unidade Central , possuindo, como todo periférico, registros internos acessados da maneira usual (7).

Surge daí um primeiro problema, que é o de evitar que a Unidade Central em operação normal, acesse esses registros, o que, evidentemente, não possui significado, e pode ocasionar "danos" à operação. Se torna então necessário "proteger" esses registros contra acesso indevido. Na medida em que o Carregador deve poder acessá-los mesmo com a máquina em execução, uma proteção completa é impossível sem o auxílio de um comando externo, diretamente ligado ao controle da interface. Uma linha de controle acionada pelo Painel da CPU foi então criada (LIB MPROG), e deverá desativar a interface sempre que essa proteção for necessária.

Foram então definidos 10 registros para a interface.Para tornar a interface "unibus-unidade de controle" compatível com aquela Terminal Inteligente-"unibus", que apenas acessa palavras (16 bits), não haverá acesso por byte, no entanto, por medida de economia de conectores os dados serão transmitidos em 8 bits, ou seja, os demais 8 bits da barra de dados não possuem significação para a interface.

Esses registros estão intimamente ligados às funçoes do carregador que expusemos anteriormente, e são os seguintes:

RØ (E8ØØ) - registro de endereço baixo (8 bits) da memória de controle.

R1 (E8Ø2) - registro de endereço alto (5 bits) da memória de controle.

R2 (E8Ø4) - registro do byte a ser escrito na memória de controle, no endereço especificado em RØ e R1.

R3 (E8Ø6) - registro de comparação baixo (8 bits).

R4 (E8Ø8) - registro de comparação alto (1 bit).

R5 (E8ØA) - registro dos modos de execução:

UNIBUS DATA (1;Ø) Ø Ø - assíncrono;
Ø 1 - interrompido;
1 Ø - micro a micro;
1 1 - ciclo de bus.

R6 (E8ØC) - comando de liberação da execução.

R7 (E8ØE) - comando de suspensão da execução.

R8 (E81Ø) -
R9 (E812) - Registros de leitura da memória de controle.

Esses dois últimos registros merecem especiais conside

rações.

Com o fim de economizar conectores o processo de leitura da memória de controle foi modificado e tornado indireto. A leitura é feita por comparação entre o byte acessado naquela memória e um segundo byte enviado pelo sistema carregador. O carregador então envia um byte para comparação e, caso não haja sucesso nessa comparação, incrementa o valor desse byte enviando-o novamente para comparação até que se consiga sucesso. Todo o processo é naturalmente, "transparente" ao operador que não o saberia distinguir da leitura direta.

Um esquema simplificado do controlador da memória de microprograma é apresentado a seguir.

É preciso ainda salientar que o "flag" indicativo da execução é interno ao sistema. É setado caso o operador comande a execução e o comando tenha sido transferido sem erros pela barra de entrada e saída. Do mesmo modo, o "flag" é resetado quando do comando de parada.

Outro ponto que se deve mencionar é aquele referente à tentativa de transferência com "unibus" ocupado pela UCP. Nesse caso, não será evidentemente, possível efetuar a transferência e o sistema sinalizará a ocorrência. Decorre desse fato que, nessas circunstâncias, não será possível ao carregador comandar a Unidade de Controle, cabendo então ao operador fazê-lo manualmente através dos controles do painel . Quando a situação tiver sido contornada, a UCP estará nova

mente ao alcance do carregador.

4.4 - Definição do Software

A atividade do operador consiste basicamente (em relação ao carregador), em enviar comandos e escrever comentários.

Definimos então os comandos através de mnemônicos e os distinguimos dos comentários fazendos preceder por um "*" (8).

As operações se dão normalmente segundo o seguinte esquema:

- comentários (data, operador, estágio da depuração, procedimento adotado, observações, etc.);
- verificação de pastilhas danificadas;
- Carga do microprograma (cartão, TTY, K7);
- modo de execução (assíncrono, micro a micro, interrupção comandada pelo endereço da micro em execução, ciclo do bus);
- comando de execução (e endereço inicial de execução);
-

.

. ... (comandos)

.

- guardar microprograma atual na fita K7;

- comentários finais.

A recepção dos comandos se faz então sob o seguinte protocolo:

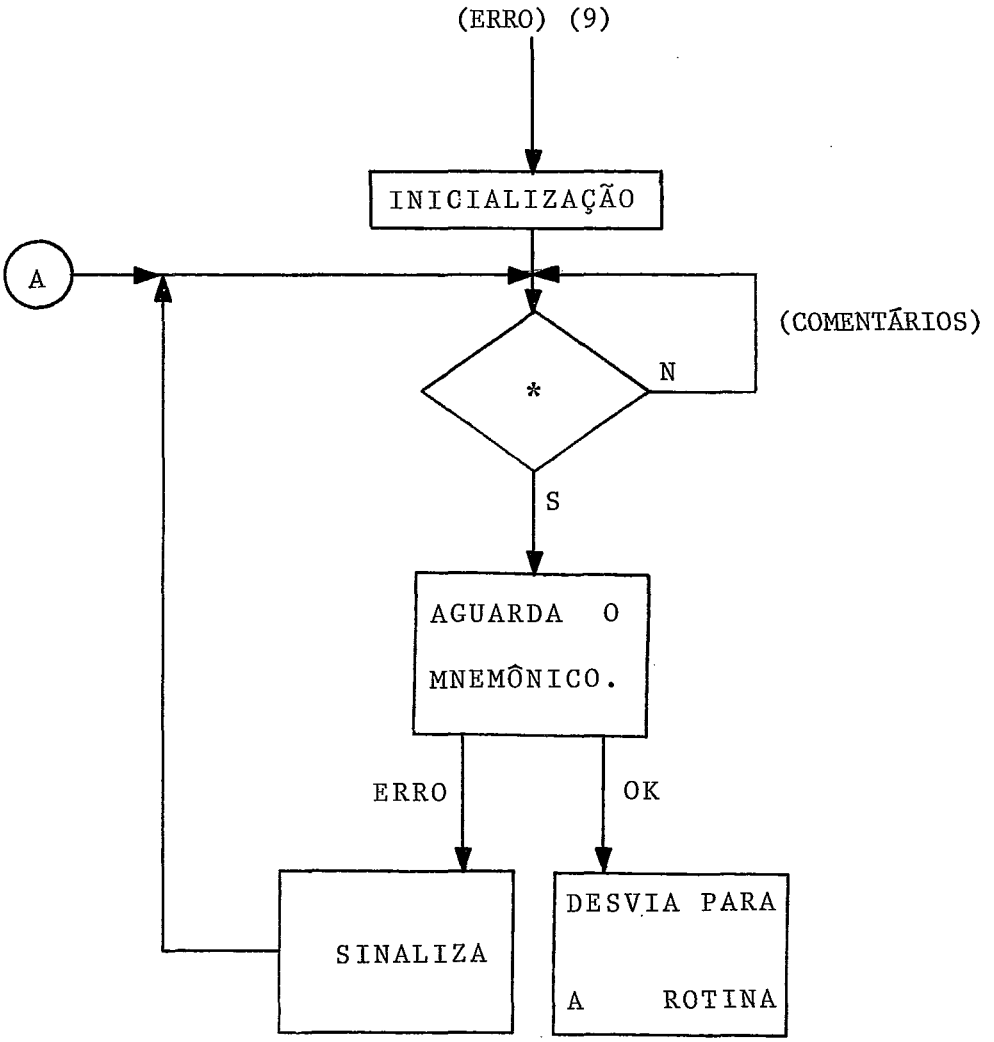


FIG 11

Mostraremos duas rotinas do sistema que o exemplificam satisfatoriamente:

A transferência de dados, de cartão perfurado para a memória de controle se faz da maneira exposta abaixo:

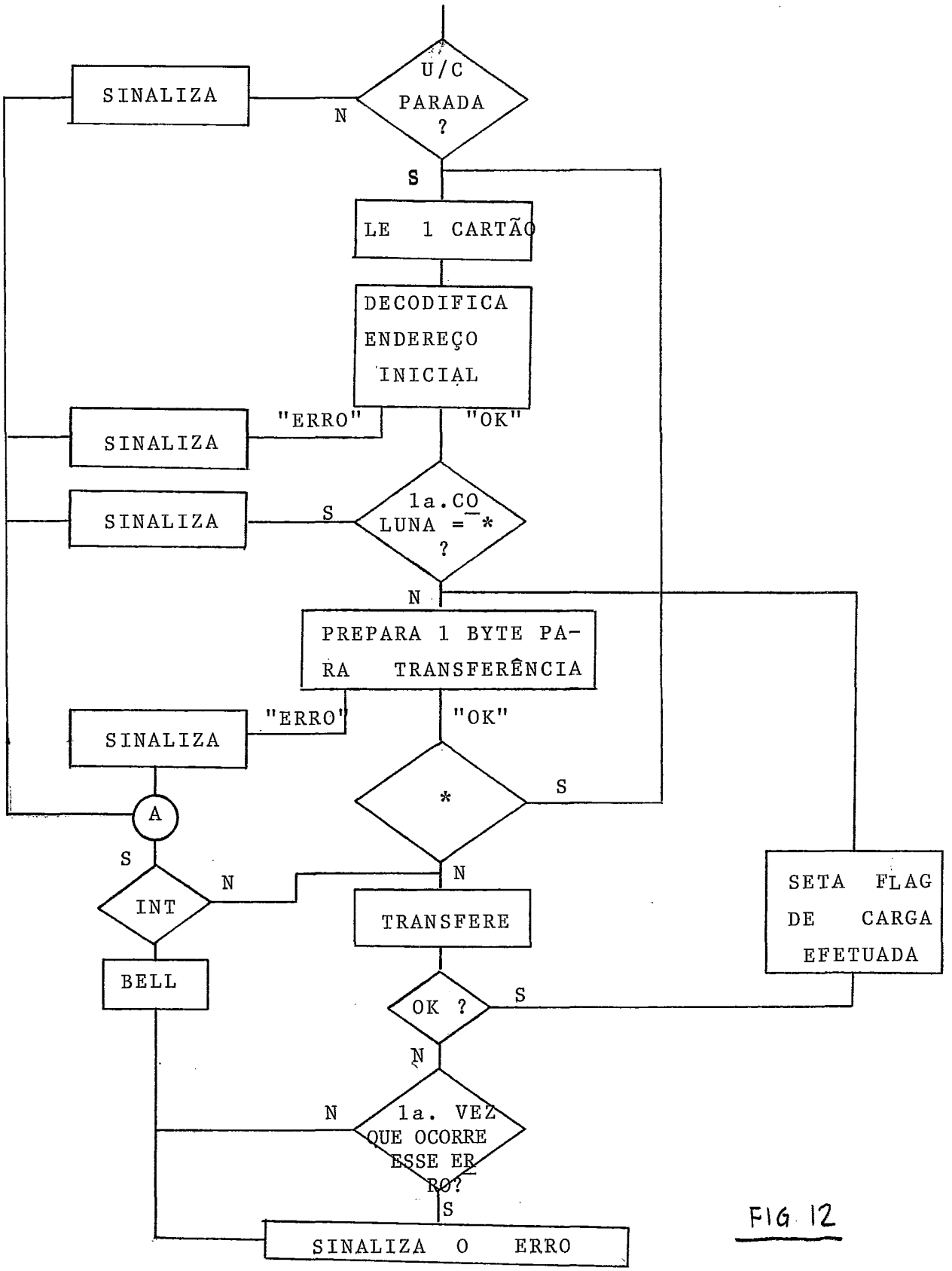


FIG 12

Como se pode observar, o sistema insiste na transferência do byte quando ocorre um erro. O motivo deste procedimento é claro: manter as condições de erro nas linhas de maneira a permitir ao operador a observação e correção do mesmo. Desejando desistir do processo, o operador interrompe acionando qualquer tecla da TTY, o "loop" será então abortado.

Os erros que podem ocorrer na transferência se referem ao não cumprimento do protocolo da barra de E/S ("unibus") em cada um dos acessos possíveis na interface.

Durante a decodificação os erros possíveis são aqueles já mencionados:

- especificação do endereço inicial;
- especificação do byte hexadecimal;
- formatação.

As demais funções de transferência funcionam segundo o mesmo princípio e não as exporemos aqui (10).

Todas as funções que transferem dados para a memória de controle se utilizam de uma rotina especial de transferência que procura "estabilizar os erros" (pelo motivo exposto) que venham a ocorrer.

A estrutura da rotina é a seguinte:

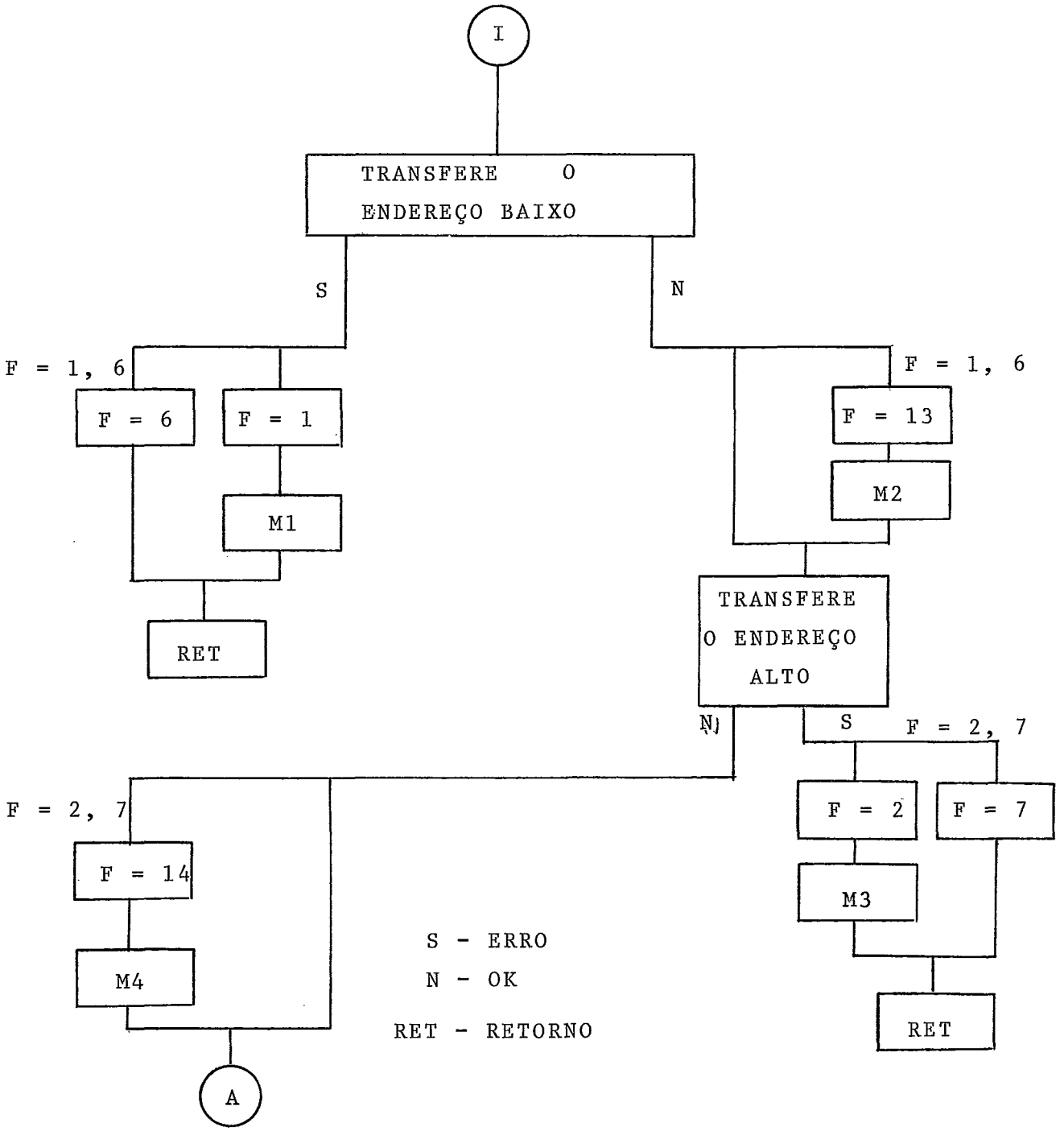
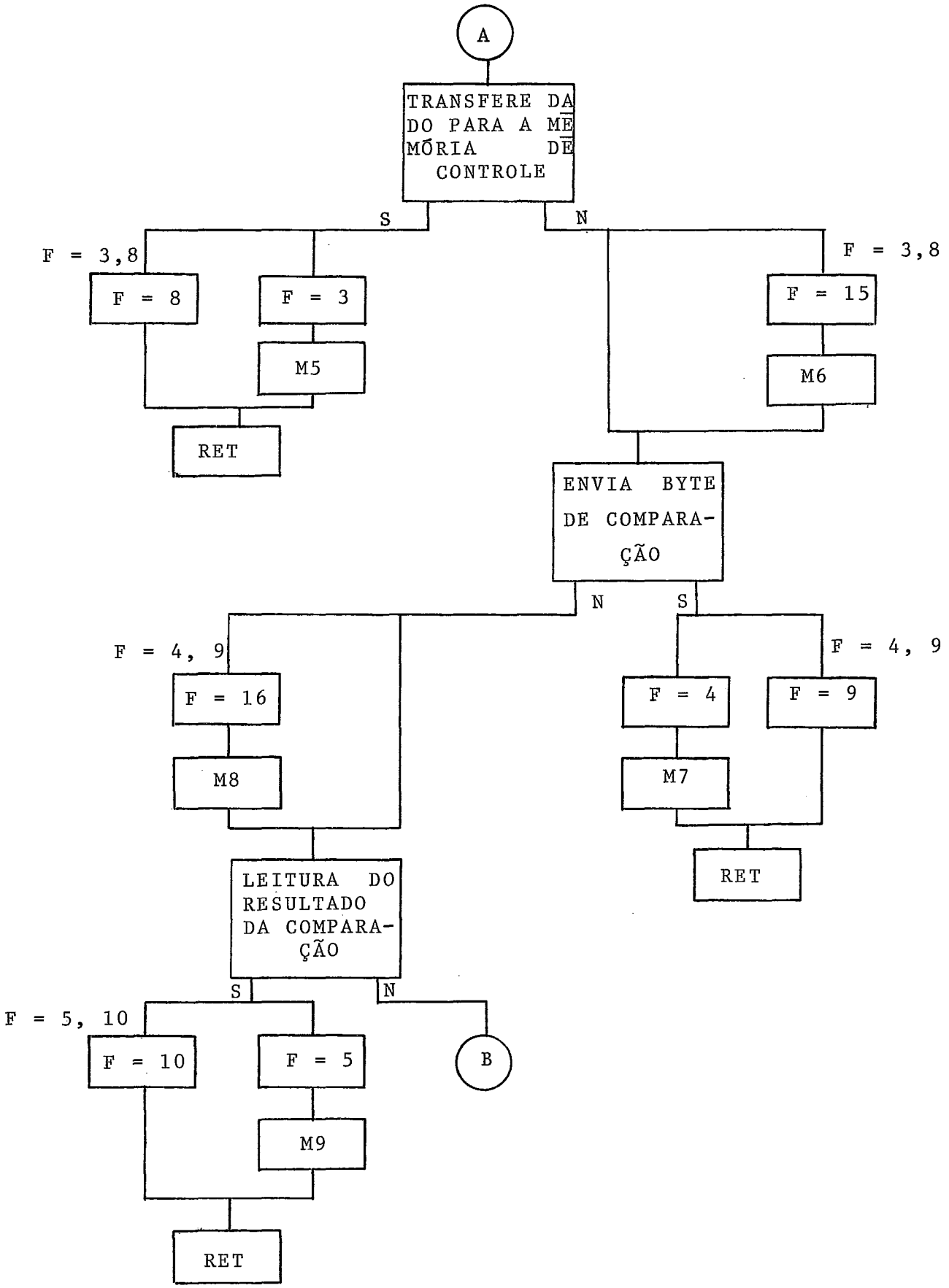
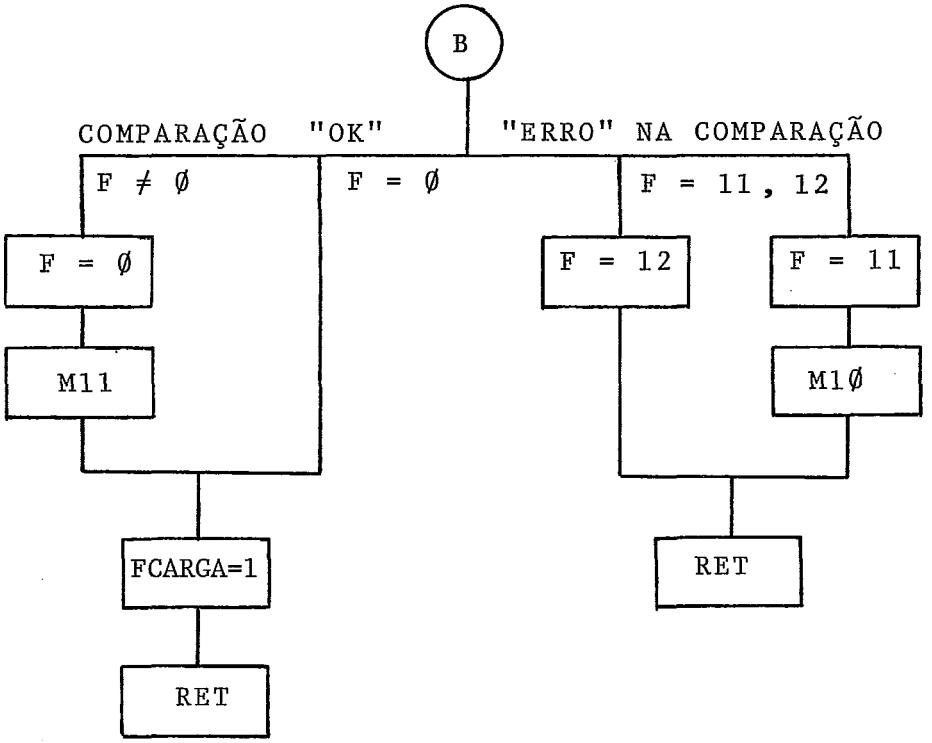


FIG. 13


A
TRANSFERE DADO PARA A MEMÓRIA DE CONTROLE
S
N
F = 3,8
F = 8
F = 3
M5
RET
F = 3,8
F = 15
M6
ENVIA BYTE DE COMPARAÇÃO
N
S
F = 4, 9
F = 16
M8
F = 4, 9
F = 4
F = 9
M7
RET
LEITURA DO RESULTADO DA COMPARAÇÃO
S
N
F = 5, 10
F = 10
F = 5
M9
RET
B



Nessa rotina "F" define o "estado de erro" na mesma que, como se depreende do diagrama, são os seguintes:

F = Ø - transmissão sem erro.

F = 1 - primeiro erro na escrita do registro de endereço baixo (7:0).

F = 2 - primeiro erro na escrito do registro do endereço alto (12:8).

F = 3 - primeiro erro na escrita da memória de controle.

F = 4 - primeiro erro no envio do byte de comparação.

F = 5 - primeiro erro durante a leitura do resultado da com-

paração.

F = 6 - já ocorreu erro nº 1.

F = 7 - já ocorreu erro nº 2.

F = 8 - já ocorreu erro nº 3.

F = 9 - já ocorreu erro nº 4.

F =1Ø - já ocorreu erro nº 5.

F =11 - primeira vez que a comparação não tem sucesso.

F =12 - já ocorreram comparações sem sucesso neste acesso.

F =13 - cessou erro nº 1.

F =14 - cessou erro nº 2.

F =15 - cessou erro nº 3.

F =16 - cessou erro nº 4.

(A ocorrência "cessou erro nº 5", se dá por exclusão).

As sinalizações por mensagens (Mi) ao operador se dão apenas nas transições de estados, visto que não é adequado permanecer sinalizando a mesma ocorrência além de que esse procedimento tornaria o "loop" demasiado lento, apenas um "bell" é emitido para sinalizar o "loop".

Por fim a variável "FCARGA" sinaliza (para o sistema) uma transferência bem sucedida.

Os flags de 13 a 16 embora desnecessários em princípio (poderiam ser iguais a zero) foram adicionados apenas para forçar a mensagem Nº 11, que reforçaria para o operador

a indicação de que a transferência foi realizada corretamente.

4.5 - Conclusão

O carregador de microprograma foi o sistema definido para se utilizar na depuração da U/C e dos microprogramas na própria máquina e nas condições mesmas em que ela operará por fim.

Evidentemente não se propõe a resolver sozinho esse problema. É apenas o mecanismo que se utilizará para transferências de dados e controle do modo de execução. Todos aqueles sinais e registros das diversas unidades da máquina que devem ser examinados lhe fogem ao controle e serão observados com os equipamentos usuais.

Outro fato a mencionar é a possibilidade de se implementar outras funções no sistema sempre que surja essa necessidade. Esse acréscimo (ou modificações) são efetuados através do SOS (Sistema Operacional de Simulação) desenvolvido no NCE/UFRJ para criação de software para o Terminal Inteligente.

1) - A gravação das PROMs será feita num programador de PROMs desenvolvido no NCE/UFRJ;

2) - O Terminal Inteligente é um microcomputador desenvolvido no NCE (Vide - "TI - Manual do Usuário" - NCE/UFRJ).

3) - Por motivos de economia de conectores a comunicação entre o Terminal Inteligente e a UCP se faz através de 8 bits de dados;

4) - Para especificação de 8 KBytes são necessários 13 bits;

5) - O décimo bit especifica a segunda palavra da microinstrução e seu controle é automático;

6) - Para uma explicação detalhada dos processos vide ADRIANO JOAQUIM DE OLIVEIRA CRUZ - "Projeto de uma UCP de Médio Porte - Unidade de Controle" - Tese/COPPE-77);

7) - Para uma explicação detalhada do "unibus" vide JÚLIO SALEK AUDE - "Projeto de uma UCP de Médio Porte - Unidade de Entrada e Saída" - Tese/COPPE-77);

8) - Os mnemônicos foram definidos com dois (2) caracteres:

FUNÇÃO	MNEMÔNICO	FUNÇÃO	MNEMÔNICO
1	LC	8	MH
2	LE	9	MM
3	MP	1Ø	MC
4	/G	11	MA
5	/C	12	CT
6	EX	13	DP
7	PA	14	VP

Como nesse caso existirão mais comentários que coman-

dos, é natural inverter a ordem usualmente adotada.

9) - Botão de Interrupção do Terminal Inteligente;

1Ø) - Os manuais técnicos e as listagens dos programas podem ser encontrados no NCE/UFRJ.

5 - DEPURADOR PROGRAMÁVEL DE CIRCUITOS DIGITAIS

5.1 - Introdução

A experiência tem demonstrado que a etapa de depuração (lógica e elétrica) de um circuito digital é um dos momentos críticos de um projeto, principalmente quando não se dispõe de equipamentos específicos numa medida satisfatória e de procedimentos organizados e padronizados que não tornem essa atividade uma tarefa "artesanal", diferente das demais fases do projeto.

Várias são as dificuldades que se apresentam, e resumidamente podemos considerar:

a) - O controle das entradas do sistema

Que comumente exige chaveamento extenso e disperso (1) pelo circuito, tanto em relação à níveis lógicos estáveis (no período considerado) com que se deve, num determinado momento, fixar algumas das entradas do sistema, como em relação a transições de níveis e pulsos de duração determinada com que se deve excitar outras entradas.

b) - O controle das saídas do sistema

Usualmente pelo menos uma parte das saídas do sistema podem ser analisadas sem conectá-las ao dispositivo externo que o projeto prevê.

Embora em tese isso sempre possa ser feito em relação à quaisquer saídas, a prática limita a importância e a significação desse procedimento em muitos casos.

Quando se tem em vista que depurar não é examinar todas as possibilidades (o que nunca é feito) e que frequentemente esse(s) dispositivo(s) externo(s) têm seu funcionamento limitado a uma "escala de tempo" própria, percebe-se que algumas restrições surgirão no processo de depuração.

Controlar as saídas de um sistema é observá-las e aos efeitos que causa no dispositivo que elas controlam.

A visualização desses pontos é comumente feita através de equipamentos especializados, a maior parte deles de introdução recente no mercado.

c) - O controle do próprio processo de depuração

Um ponto básico e, na maioria dos casos, insatisfatoriamente resolvido.

Embora esse ponto seja fundamentalmente dependente da infraestrutura do laboratório. alguma coisa pode ser feita para minimizar suas consequências. Uma delas é unificar a documentação que normalmente é gerada de maneira esparsa e submetê-la a uma certa padronização.

Uma proposta detalhada de tal método foge ao escopo desse trabalho, mas tentaremos mostrar que o sistema a

qui apresentado,apresenta algumas das características necessárias ao mesmo.

Convém lembrar que, diferentemente do teste de circuitos, a depuração é um processo iterativo que exige um "operador qualificado", que conheça o sistema nos seus mínimos detalhes, que defina estágios a serem vencidos, e que mude seu "algorítmo de depuração" a cada estágio, fazendo, entre eles, as correções no circuito que se fizerem necessárias.

Normalmente, durante a depuração, certas configurações de níveis lógicos, transições de níveis e pulsos (controles) são aplicadas às entradas do sistema, repetidas algumas vezes e após o exame das saídas relevantes, e a análise do funcionamento são modificadas para testar-se um outro aspecto e assim por diante até que o sistema tenha sido observado para um conjunto expressivo das configurações possíveis.

Todos esses aspectos nos levaram a crer que um sistema programável, "ON-LINE", de depuração seria um instrumento valioso nessa etapa dos projetos.

O sistema foi definido com uma motivação essencialmente prática e visando, através da simplicidade, se apresentar de forma a minimizar a "rejeição natural" que poderá se desenvolver nos engenheiros do ramo, habituados que estão a um procedimento muito diferente.

Além disso, o sistema foi desenvolvido visando resolver uma necessidade criada pelo projeto da UCP, o que nos

levou a utilizar o Terminal Inteligente, também desenvolvido no NCE/UFRJ, para sediar o sistema.

Com isso algumas restrições, principalmente no tocante à "velocidade do sistema", limitaram seu alcance, no entanto, para quase todos os propósitos básicos, essa restrição não se mostra de grande importância.

Certamente, o sistema não gerará configurações de entrada nem mostrará as saídas muito rapidamente, além de não poder se sincronizar aos circuitos além de certa velocidade, mas isso não impede que sua utilização seja poderosa nas fases em que o circuito em depuração não precisar ser testado sob as condições em que normalmente operará.

A troca de informação entre o sistema e o circuito em depuração é lenta, mas, para boa parte dos casos, satisfaz as exigências. Além disso, considere-se que a "velocidade do operador" de analisar e modificar as configurações é relevante e não pode ser aumentada. Essa relevância é, naturalmente, relativa. Não é essa característica que limita a velocidade com que, para um teste decisivo se deve excitar os circuitos. A limitação se manifesta mais claramente na impossibilidade de se gerar testes "longos" e complexos, na medida em que um desvio ou erro na resposta esperada não permite, nesse caso, a identificação da etapa intermediária em que o mesmo se deu. Ocorre então que, um determinado processo pode

requerer um número elevado de "cortes", de etapas intermediárias que irão exigir análises respectivas e tornar proporcionalmente lento todo o processo.

(1) - Devido principalmente às técnicas de montagem de um circuito não se consegue uma subdivisão do sistema tão flexível como no caso do software o que leva, muitas vezes, à montagem de uma parte do circuito substancialmente maior que aquela em depuração num dado momento.

5.2 - Definição do Sistema

Na definição do sistema, tanto em "potencialidade" - (software) - como no seu "dimensionamento - (hardware) - o critério de "sistema mínimo" foi adotado. A razão maior para esse procedimento foi a expectativa de que sua utilização prática sugerisse modificações substanciais.

A FIG. 1 mostra o esquema de acoplamento entre as várias partes do sistema:

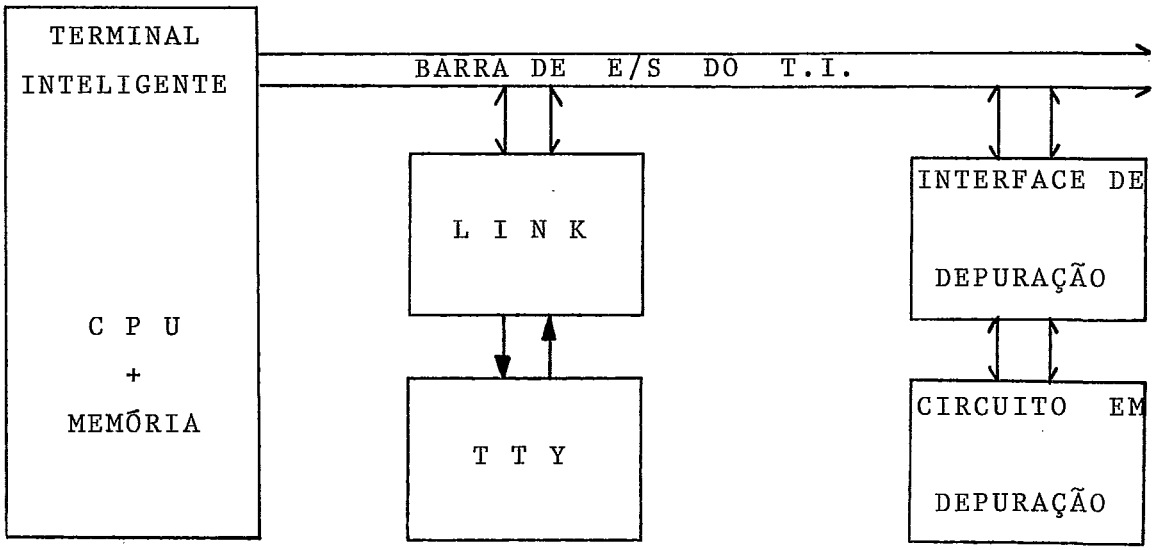


FIG. 14

A operação do sistema se entende então facilmente: o operador digita o programa de depuração na TTY, o sistema o recebe via link (gerando na TTY um documento do procedimento adotado, que deve ser tornado claro pelo operador através de descrições e comentários) e oferece ao operador algumas facilidades (descritas adiante), além de fazer uma crítica do programa, que só será executado caso esteja "correto".

O programa de depuração referencia implícita ou explicitamente os registros da "Interface de Depuração", ou seja, o "Programa de Depuração" é executado "sobre" a Interface. Para tanto, é necessário que o operador tenha providenciado as ligações necessárias entre o circuito em depuração e a Interface (Painel).

Um exemplo completo é apresentado na seção 5.6. O sistema pode ser dividido então em três (3) partes:

a) - O MONTADOR - que recebe, critica, e monta as instruções do programa, além de oferecer algumas facilidades ao operador.

b) - O INTERPRETADOR - que executa o programa e trata das interrupções e dos comandos oferecidos.

c) - A INTERFACE - que troca sinais elétricos com o circuito em depuração.

O programa, como se vê, é interpretado. Essa escolha se deve, basicamente, à maior simplicidade de projeto e à flexibilidade que a solução oferece, haja visto que a velocidade de execução do programa de depuração não é, nesse caso, um fator muito importante.

Na definição do sistema adotamos ainda os seguintes procedimentos:

- toda instrução possui um "label" associado: o número de ordem da instrução;

- o endereço de carga dos programas de depuração é fixo.

O primeiro se deve à simplificação do processo de montagem e "dump" do programa, além de que, pelas razões já expostas, se mostra um procedimento útil ao usuário desse tipo de sistema.

O segundo explica-se pelo desinteresse do programador quanto ao endereço de carga de seu programa, que é então escolhido segundo as conveniências do sistema.

5.3 - Descrição da Interface

É através dessa interface que o sistema se acopla e comanda o circuito em depuração.

É constituída de seis (6) partes:

1) - Três (3) registros de 8 bits que fornecem níveis estáveis;

2) - Um (1) registro de 8 pulsos independentes de duração selecionada;

3) - Três (3) sensores de níveis, de 8 bits cada um;

4) - Um (1) registro de "status", onde se armazenam o "estado anterior" de 8 linhas e que, para cada uma dessas linhas, pode armazenar um (1) de cinco (5) estados:
   1 - nenhuma transição;
   2 - uma transição positiva;

3 - uma transição negativa;

4 - uma transição positiva seguida de outras transições;

5 - uma transição negativa seguida de outras transições.

5) - Uma linha de sincronismo por transição que também é utilizada para medidas de níveis lógicos (Ø, 1,abetto).

6) - Uma ponteira lógica.

Um esquema simplificado do circuito, poderia ser o apresentado à seguir (FIG. 2).

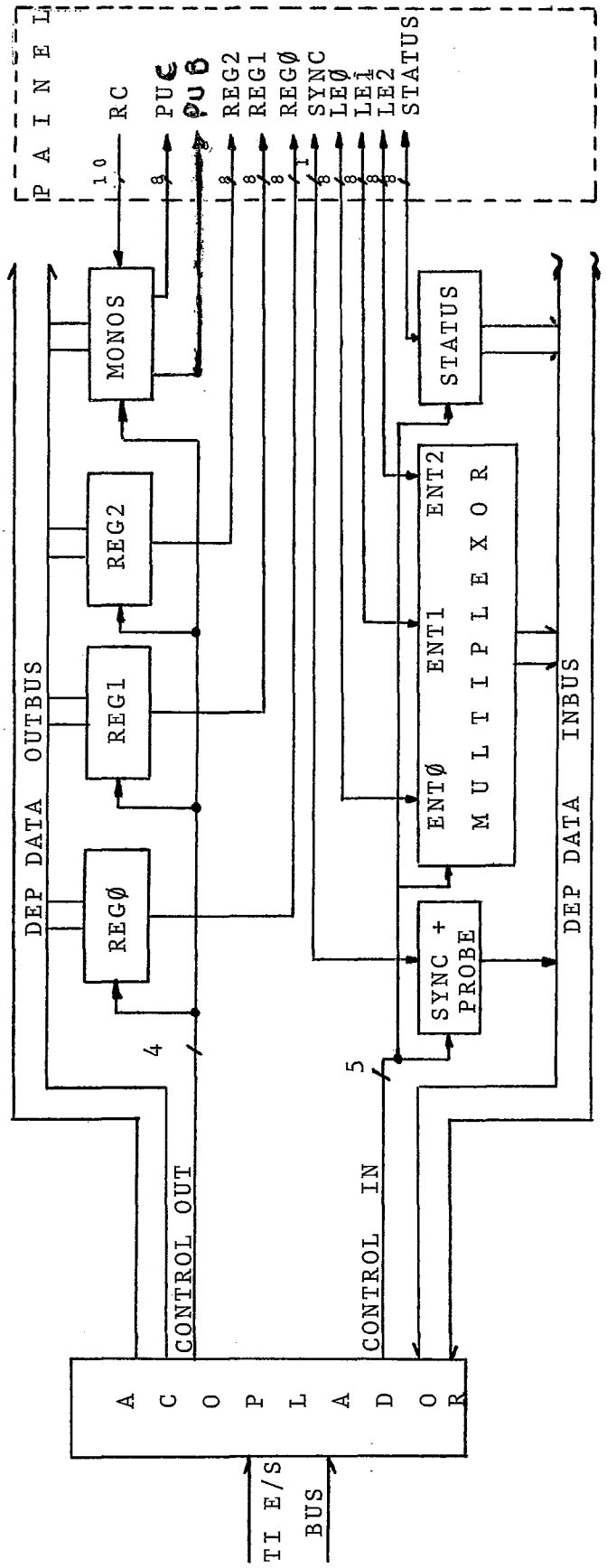
A interface de depuração, como se pode observar, foi dimensionada para uma capacidade média de ligações, embora possa ser expandida, se necessário.

Duas observações se fazem necessárias:

1) - Exceto no caso da mesma configuração, não é possível carregar os registros de saída, simultaneamente;

2) - As entradas não são armazenadas, não é pois, possível uma leitura "simultânea", há uma defasagem de aproximadamente 20 us entre as leituras de cada sensor.

As duas características não degradam no entanto a performance do sistema, pois na maioria dos casos essa simultaneidade não é indispensável. É conveniente, então, minimizar o hardware necessário.



RC - 5 x 2 linhas das entradas RES e CAP dos monoestáveis (os outros três monos são fixos em 4µs).

PUC - Saídas "Q" DOS MONOS.

PUB - Saídas "$\overline{Q}$" DOS MONOS.

REGØ, 1, 2 - SAÍDAS DOS REGISTROS.

LEØ, 1, 2 - ENTRADAS MULTIPLEXADAS.

SYNC - LINHA DE SINCRONISMO POR TRANSIÇÃO E "LOGIC PROBE".

STATUS - Linhas de DETEÇÃO DE "STATUS".

FIG. 15

Uma outra característica é o fato do depurador ser uma máquina TTL, projetada para trabalhar sobre circuitos dessa família, o "fan-out" de cada saída é o padrão, como também o "fan-in" das entradas, não havendo reduções adicionais (exceto para a entrada "SYNC").

5.4 - Arquitetura do Software

Por arquitetura do software, entendemos aqui questões como os protocolos de entrada dos programas, dos comandos e das interrupções, as facilidades oferecidas ao operador e do nível das mensagens enviadas pelo sistema.

Adotamos então cinco (5) critérios básicos:

1) - Especificação dos programas de depuração semelhante àquela adotada na programação ASSEMBLER para o Terminal Inteligente;

2) - Crítica do programa a cada instrução enviada de modo a permitir correções a cada passo;

3) - Sinalização de todos os erros ocorridos através de mensagens enviadas à TTY.

4) - Comandos especificados por um (1) caracter;

5) - Sistema com interrupção, de modo que a execução do programa possa ser interrompida no final de cada instrução, possuindo o operador nessa situação, diversos comandos disponíveis.

As facilidades que julgamos convenientes são as seguintes:

1) - Durante a programação:

a - o cancelamento da última instrução;

b - o cancelamento de todo o programa;

c - comentários.

2) - No final da programação:

a - cancelamento da última instrução (no caso só pode ser "FIM") e a continuação da programação;

b - o cancelamento de todo o programa;

c - "dump" do programa no formato fonte;

d - execução assíncrona ou "passo a passo" com a especificação do endereço de execução, ou seja, do número de ordem da primeira instrução a ser executada.

e - comentários.

3) - Durante a execução (por interrupção):

a - continuação da execução;

b - executar do início do programa (ØØØ);

c - executar a partir de uma determinada instrução;

d - modificar o modo de execução;

e - voltar ao montador para extender o programa;

f - cancelar todo o programa;

g - comentários.

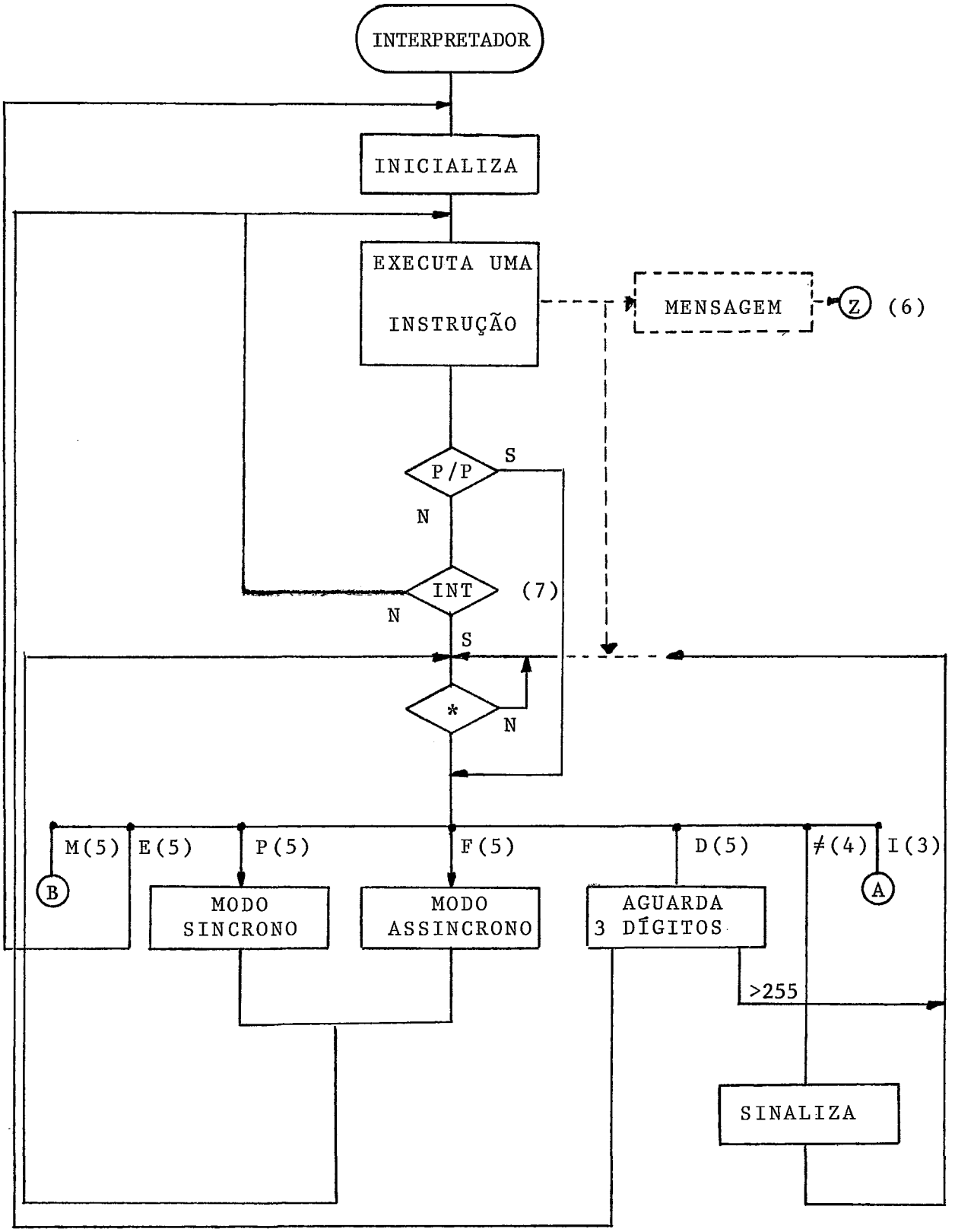

INTERPRETADOR
INICIALIZA
EXECUTA UMA INSTRUÇÃO
MENSAGEM
Z
(6)
P/P
S
N
INT
(7)
N
S
*
N
M(5)
E(5)
P(5)
F(5)
D(5)
≠(4)
I(3)
B
A
MODO SINCRONO
MODO ASSINCRONO
AGUARDA 3 DÍGITOS
>255
SINALIZA

FIG. 17

1) - Botão de interrupção do Terminal Inteligente (desvia para a posição /ØØ38);

2) - Ø - comando "INICIALIZE";
1 - comando "CANCELE ULTIMA INTRUÇÃO DO PROGRAMA";

3) - Comandos possíveis no final da montagem:

I - INICIALIZE (Cancela todo o programa);
D - DUMP do programa no formato fonte;
P - EXECUÇÃO PASSO A PASSO (síncrona);
E - EXECUÇÃO ASSÍNCRONA;
M - MONTE. Estende o programa.

4) - Comando Inválido.

5) - Comandos possíveis de interrupção da execução:

I - INICIALIZE (Cancela todo o programa);
D - DESVIO PARA INSTRUÇÃO NNN (NNN $\leq$ 255);
E - EXECUTE NOVAMENTE A PARTIR DA INSTRUÇAO ØØØ;
P,F - SELECIONA MODO DE EXECUÇÃO;
M - MONTE. Retorna ao montador para estenção do programa.

6) - Caso haja um desvio para uma instrução inexistente, ou seja, cujo "label" seja maior que o tamanho do programa.

7) - O sistema é interrompido acionando-se qualquer tecla da TTY.

5.5 - Conjunto de Instruções

O sistema que definimos é específico, logo o conjunto de instruções deve atender a essa especificidade, além de evitar as complexidades desnecessárias que surgem num "set" de instruções de uma máquina "comum".

Assim, o único modo de endereçamento necessário é o IMEDIATO posto que um "programa de depuração" precisa apenas fazer referências diretas aos registros da Interface de Depuração (que chamaremos doravante de registros do depurador), ou seja, aos pontos do circuito em depuração que estão sob controle do operador.

Um conjunto mínimo de instruções foi então definido e implementado procurando dar flexibilidade e simplicidade à programação.

Note-se, no entanto, que o operador, face a um problema específico, poderá criar instruções adicionais que melhor o satisfaçam. Naturalmente, esse acréscimo não pode ser efetuado no próprio Depurador. A compilação do sistema é feita através do Sistema Operacional de Simulação (SOS) desenvolvido no NCE/UFRJ com vistas à criação de software para o Terminal Inteligente e que "roda" no BURROUGHS B-6700, da UFRJ.

A definição de um conjunto de instruções para uma "máquina" assim específica deve mais seguir uma orientação baseada na experiência adquirida das necessidades do processo do que

a considerações formais.

Basicamente, são quatro (4) as características gerais que o sistema deve atender:

1) - Excitar o circuito e captar suas respostas;

2) - Fazer comparações, efetuar desvios e sincronizar;

3) - Possuir facilidades na alterações dos padrões de excitação (os registros de saída);

4) - Oferecer facilidades na execução de programas.

A descrição das instruções mostrará mais claramente esse ponto.

Nessa descrição os registros de saída são referenciados como R (Ø, 1, 2); as entradas como E(Ø, 1, 2) e os tipos de sincronismos como S (Ø, 1, 2).

Foram definidos onze (11) tipos de instruções que descreveremos no formato:

característica que a motivou - mnemônico - descrição.

TIPO Ø - ZERO OPERANDOS

São instruções que não necessitam fazer quaisquer referências.

(4) - NOP - No-operation.

(4) - EXEC - Executa novamente o programa (1).

(4) - WAIT - Interrompe a execução, sinaliza ao operador e aguarda um dos comandos especificados.

(1) - STAT - "status". Escreve na TTY o estado das 8 linhas que registram transições.

(4) - PARE - "stop". Termina a execução e aguarda um comando.

(3) - CLEAR - Zera simultaneamente os três (3) registros de saída.

(1) - READR - Escreve na TTY o valor dos três (3)registros (2).

(1) - READE - Escreve na TTY o valor das três (3) entradas (2).

(4) - BIP - Envia um "bell" à TTY.

(4) - PP - A partir dessa instrução o programa será executado instrução a instrução (3).

(4) - RUN - A partir dessa instrução o programa será executado assincronamente.

(1) - PROBE - Escreve na TTY o nível lógico da entrada S (Ø, 1, A (aberto), P (Pulsando)).

(4) - FIM - "end". Não é, no entanto, uma pseudo. É montada e executada como "PARE".

TIPO 1 - UM OPERANDO (E) OU (R) OU (S)

(3) - INC - Incrementa o registro (R).

(3) - DEC - Decrementa o registro (D).

(3) - SHE - "shifta" o registro (R) uma posição à esquerda (9).

(3) - SHD - "shifta" o registro (R) uma posição à direita (9).

(3) - RRE - "rotate" o registro (R) uma posição à esquerda (9).

(3) - RRD - "rotate" o registro (R) uma posição à direita (9).

(1) - LEE - Escreve na TTY o valor da entrada (E). (2)

(1) - LER - Escreve na TTY o valor do registro(R). (2)

(2) - SYNC - sincroniza o programa com uma transição externa do seguinte modo: (6)

Sincronizado se:

S = Ø - transição negativa;

S = 1 - transição positiva;

S = 2 - qualquer transição.

TIPO 2 - UM OPERANDO, DOIS (2) CARACTERES HEXADECIMAIS (/HH).

(1) - PULSO - Aciona os monoestáveis segundo a "máscara" (/HH).

(1) - WRITE - Escreve o operando (/HH) simultaneamente nos três (3) registros.

TIPO 3 - UM OPERANDO, TRÊS (3) CARACTERES NUMÉRICOS (NNN) ONDE NNN $\leq$ 255. (4)

(2) - GOTO - Desvio incondicional para a instrução de número (NNN).

As seis (6) instruções que se seguem são de desvio condicional sobre o "flag" interno gerado por uma instrução de "COMPARE" (vide, TIPO 7).

(2) - JI - Desvia se IGUAL;
(2) - JD - Desvia se DIFERENTE;
(2) - JG - Desvia se MAIOR;
(2) - JL - Desvia se MENOR;
(2) - JGI - Desvia se MAIOR OU IGUAL;
(2) - JLI - Desvia se MENOR OU IGUAL.

As duas (2) instruções a seguir, desviam sobre o "flag" interno gerado pelas instruções de "SHIFT" e "ROTATE" (Carry).

(2) - JTC - Desvia se CARRY = 1.

(2) - JFC - Desvia se CARRY = Ø.

TIPO 4 - DOIS OPERANDOS: (R) OU (E); (/HH)

(1) - MVI - Move (/HH) para (R).

(1) - MVII - Move (/HH) complementado a um para (R).

(2) - SW - Sincroniza o programa com a deteção na entrada (E) da palavra (/HH) (5).

(2) - INTE - Lê a entrada (E) e caso seja igual a (/HH), sinaliza ao operador e interrompe a execução, caso contrário prossegue.

(2) - INTD - Lê a entrada (E) e caso seja diferente de (/HH) sinaliza os dois valores ao operador e interrompe a execução. Caso contrário prossegue.

TIPO 5 - DOIS OPERANDOS; (R), (NNN($\leq$ 255))

(2) - JIZ - Desvio condicional sobre o registro (R) para a instrução (NNN) (4). A condição é que o registro esteja zerado.

(2) - JDZ - O mesmo que JIZ, a condição é que o registro não esteja zerado.

TIPO 6 - NÚMERO INDEFINIDO ($\leq$ 255) DE OPERANDO NO FORMATO: (R); (/HH, ..., /HH,.)

(3) - SQ - Gerador de Caracteres.
Os caracteres (/HH), são aplicados ao registro sucessivamente (7).

TIPO 7 - DOIS OPERANDOS: (E) OU (R); (E) OU (R)

(3) - MOV - Move o registro $(R)_1$ para o registro $(R)_2$

(2) - CEE - Compara a entrada (E_1) com a entrada (E_2). O resultado da comparação é armazenado internamente em uma de três possibilidades: =, < , >.

(2) - CES - Compara a entrada (E_1) com a saída (R_2). O resultado é armazenado como em "CEE".

(2) - CSS - Compara o registro de saída (R_1) com o registro (R_2). O resultado é armazenado como em "CEE".

(3) - SWAB - Troca os valores dos registros $(R)_1$ e $(R)_2$.

(3) - MVES - Move o valor da entrada $(E)_1$ para o registro $(R)_2$.

TIPO 8 - UM OPERANDO (NNN)

(3) - RN - Faz $(R)_1 = N_1$; $(R)_1 = N_2$ e $(R)_3 = N_3$.

TIPO 9 - SEIS OPERANDOS: (S); (/HH); (/HH); (/HH) ; (NNNN); (NN).

ALCT Analizador lógico "triggado".

- (S) - especifica o tipo de clock (= SYNC);
- $(/HH)_{1,2,3}$ - especificam a palavra (24 bits) de trigger.
- (NNNN) - especifica o "delay" em quatro dígitos.
- (NN) - especifica o número de palavras (24 bits) a serem listados na TTY. (8)

TIPO 1Ø - TRÊS OPERANDOS: (S); (NNNN); (NN).

ALST Analisador lógico não "triggado"-

Os operandos têm o mesmo significado que em "ALCT"(8).

Estas duas últimas instruções não obedecem à motivação que expusemos e na realidade dificilmente farão parte de um programa maior. Sua inclusão se deve à grande utilidade que um analisador lógico (mesmo lento, no nosso caso cerca de 500 Hz) tem para a depuração de um circuito digital.

Observe-se que a preocupação básica não é criar formal

mente um conjunto "completo" de instruções, mas, dentro de certos limites, oferecer ao operador, mesmo que redundantemente, um conjunto de instruções que tornem a tarefa de criar um programa de depuração a mais simples e atrativa possível.

Note-se a ausência de instruções lógicas e aritméticas. Além disso, facilidades usuais dos assemblers tais como reserva de áreas, (DS) especificação de constantes (DC), definição de endereços específicos de montagem (ORG), etc., não são fornecidos por serem desnecessários nesse sistema.

(1) - Sinaliza o início do programa (≠ GOTO ØØØ);

(2) - Em notação hexadecimal;

(3) - O operador, para continuar, envia o comando "CONTINUE" (Vide seção 5.6);

(4) - Na implementação inicial, o número máximo de instruções de um programa foi limitado à 256.

(5) - Passível de interrupção durante a execução.

(6) - Não permite interrupção durante a execução;

(7) - Na realidade uma instrução "SQ" de N caracteres (/HH) se decompõe em N instruções "SQ" (R); (/HH), podendo ser interrompida e acessada internamente por desvios.

(8) - Aguarda no máximo 64 K "clocks", após o que desiste de "triggar".

Caso a frequência do sinal de sincronismo seja superior àquela que o programa pode manipular, uma sinalização (mensagem) é enviada, ao operador e o programa é "abortado".

(9) - Comandam o "flag de carry".

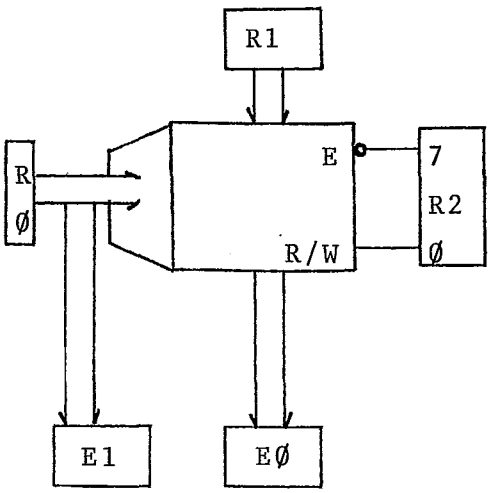
5.6 - Exemplos de Programas

Não teria sentido fazer aqui um "programa de depuração" propriamente dito, pois um programa dessa natureza não é mais do que parte do procedimento total, e encontra sua significação apenas nesse sentido. Faremos então alguns "programas de teste" como exemplo da utilização das instruções que apresentamos.

EXEMPLO 1 - Teste de uma memória RAM de 256 palavras de 8 bits: a) - EXAUSTIVAMENTE;

b) - ATRAVÉS DE DUAS (2) CONFIGURAÇÕES DE TESTE (/55 e /AA)

MAPA DE LIGAÇÕES



R/W - Ø - WRITE

1 - READ

E - Ø - ON

1 - OFF

a) - TESTE EXAUSTIVO SEM CONTROLE DE CURTO NA ENTRADA:

```
ØØØ     CLEAR       ZERA  RØ e R1. Modo WRITE.
ØØ1     INC   2           MODO  READ
ØØ2     CES   Ø  1        TESTA SE  EØ = R1
ØØ3     JD    Ø1Ø         DESVIA SE ERRO
ØØ4     DEC   2           MODO ESCRITA
ØØ5     INC   1           PRÓXIMA PALAVRA DE TESTE
ØØ6     JDZ   1  ØØ1      TESTA SE FIM DA PALS DE TESTE
ØØ7     INC   Ø           PRÓXIMO ENDEREÇO.
ØØ8     JDZ   Ø  ØØ1      TESTA SE "VARREU" TODOS OS ENDS.
ØØ9     PARE              SE TERMINOU PARA.
Ø1Ø     LER   Ø           EM CASO DE ERRO MOSTRA: EØ,
Ø11     LER   1                                   R1,
Ø12     LEE   Ø                                   RØ
Ø13     WAIT              ESPERA COMANDO DO OPERADOR
Ø14     FIM               FIM.
```

b) - TESTA COM CONFIGURAÇÕES /55 E /AA COM CONTROLE DE CURTO NA ENTRADA

```
ØØØ     CLEAR           ZERA RØ  MODO  WRITE
ØØ1     MVI   1, 55     ESCREVE  /55  NO  REG(1)
ØØ2     CPES  1, Ø      TESTA SE E1=RØ  (CURTOS)
ØØ3     JD    Ø15       SE NÃO FOR DESVIA:   ERRO
ØØ4     INC   2         MODO LEITURA
ØØ5     CPES  Ø, 1      TESTA SE  EQ=R1
ØØ6     JD    Ø19       SE NÃO FOR DESVIA:  ERRO
ØØ7     MVI   1, AA     2º PADRÃO:   /AA
ØØ8     INC   2         ESCREVE
ØØ9     DEC   2         LÊ
Ø1Ø     CPES  Ø, 1      TESTA SE  EØ=R1
Ø11     JD    Ø19       SE NÃO FOR DESVIA:   ERRO
Ø12     INC   Ø         PROXIMO ENDEREÇO
Ø13     JDZ   ØØ1       VERIFICA SE "VARREU" TODOS
Ø14     PARE            FIM DO TESTE
Ø15     LER   Ø         EM CASO DE CURTOS NA ENTRADA  MOS
Ø16     LEE   1         TRA  EØ  E  R1
Ø17     WAIT            ESPERA DECISÃO DO OPERADOR
Ø19     LER   1         EM CASO DE ERRO NO CONTEÚDO  MOS-
Ø2Ø     LEE   Ø         TRA  EØ,  E1;  RØ  E  R1.
Ø21     GOTO  Ø15
Ø22     FIM             FIM.
```

```
* EXEMPLO  2 - Teste da Interface de Depuração
*       LIGAÇÕES:
*       REG(Ø) = ENT(Ø)
*       REG(1) = ENT(1)
*       REG(2) = ENT(2)
*       REG. PULSOS(Q) = REG. STATUS
*       SYNC = OSCILADOR EXTERNO
*
*
ØØØ     WRITE  55       ESCREVE A MESMA CONFIGURAÇÃO  NOS
*                       TRÊS (3) REGISTROS.
ØØ1     INTD   Ø 55     INTERROMPE SE E(Ø) ≠ R(Ø)
ØØ2     INTD   1 55     INTERROMPE SE E(1) ≠ R(1)
ØØ3     INTD   2 55     INTERROMPE SE E(2) ≠ R(2)
ØØ4     RRE    Ø        CONFIGURAÇÃO(2) = /AA
ØØ5     RRE    1
ØØ6     RRE    2
ØØ7     INTD   Ø AA     TESTES PARA CONFIGURAÇÃO  /AA.
ØØ8     INTD   1 AA
ØØ9     INTD   2 AA
Ø1Ø     PULSO  FF       LIBERA TODOS OS PULSOS.
Ø11     STAT            O OPERADOR DEVE CONFERIR
*  O RESULTADO.  O CORRETO E TODAS AS LINHAS  ACUSANDO
*  "C".   TROCA ENTAO A SAIDA DO   REG. PULSO     PARA
*  TESTAR TRANSIÇOES NEGATIVAS.
Ø12     WAIT            ESPERA TROCA DE REG. PULSO
Ø13     PULSO  FF
```

```
014     STAT
015     WAIT
016     SYNC  Ø         TESTA TRANSIÇÃO NEGATIVA
017     WAIT
018     SYNC  1         TESTA TRANSIÇÃO POSITIVA
019     FIM
*   OSCILADOR EXTERNO DEVE   SER    COMANDADO      PARA
*   LIBERAR         APENAS         UM              PULSO
*   BYE
*   .
```

5.7 - Conclusão

Existem algumas características do sistema que apresentamos que são interessantes mencionar.

Em primeiro lugar, o sistema não possui o conceito de subrotina. Isso se deve à natureza "volátil" dos programas de depuração, ou seja, um dado programa só tem utilidade num determinado momento do processo e é modificado logo a seguir. Não há então necessidade de se criar arquivos (bibliotecas), não havendo então link-edição.

Em segundo lugar, como vimos, o único erro possível numa instrução de desvio é aquele em que a instrução acessada não foi definida no programa. Nesse caso o erro é detetado na execução e não na montagem. Esse procedimento se deve apenas à maior simplicidade deste método e à maior economia de memória que proporciona, visto que de uma maneira ou de outra o trabalho de se reconstruir o programa é o mesmo e, como vimos acima, os programas de depuração não possuem subrotinas.

6 - BIBLIOGRAFIA

1.1 - Microprogramming Principles and Practices
Samir Husson.

1.2 - Theory and Design of Digital Computers
Lewin

1.3 - Digital Systems
Hill/Peterson

1.4 - System Programming
J. Donovan

1.5 - Fundamental Algorithms
Knuth

1.6 - Designing with TTL Integrated Circuits
Texas Instruments Corporation

2.1 - PDP-11/70 - Processor Handbook

2.2 - PDP-11 - Periferals Handbook

2.3 - PDP-11/70 - Manuais diversos de circuitos

2.4 - Terminal Inteligente - Manual do Usuário

2.5 - Signectics, Texas - Manuais de CI's.

3.1 - Tecnologia e Produção Capitalista (1)
Ricardo de M. L. Tolipan

3.2 - A política da ciência no Brasil: uma discussão (1)
Vanya Sant'ana

3.3 - Considerações sobre a política científica e tecnológica no Brasil (1)
José Goldenberg

3.4 - Cartéis e Desnacionalização
Moniz Bandeira

3.5 - O momento decisivo para o computador brasileiro (2)
Ivan da Costa Marques

3.6 - A opção urgente: Autonomia ou Dependência Tecnológica (2) - Ivan da Costa Marques

3.7 - Uma etapa histórica desmentida (2)
Ivan da Costa Marques

3.8 - Uma política industrial de Informática (2)
Ivan da Costa Marques

3.9 - A Universidade como fator de autonomia tecnológica (2)
Wilson de Pádua Paula Filho

3.10 - Projeto G-10 - O domínio de sua tecnologia (2)
Sérgio Teixeira

(1) - Estudo CEBRAP - Nº 11;

(2) - DADOS & IDÉIAS.