LCS -

UMA LINGUAGEM DE MEDIO NIVEL

PARA MINICOMPUTADOR

EDUARDO LESSA PEIXOTO DE AZEVEDO

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENACAO DOS PROGRAMAS DE POS-GRADUACAO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSARIOS PARA A OBTENCAO DO GRAU DE MESTRE EM CIENCIAS (M.SC.)

APROVADA POR:

NELSON MACULAN FILHO
(PRESIDENTE)

YSMAR VIANNA E SILVA FILHO

JOAO LIZARDO RODRIGUES H. DE ARAUJO

RIO DE JANEIRO, RJ - BRASIL

MARCO DE 1977

AZEVEDO, EDUARDO LESSA PEIXOTO DE
LCS - UMA LINGUAGEM DE MEDIO NIVEL
PARA MINICOMPUTADOR [RIO DE JANEIRO] 1977.
VII, 66 P. 29,7 CM (COPPE-UFRJ, M.SC.
ENGENHARIA DE SISTEMAS, 1977)

TESE - UNIV. FED. RIO DE JANEIRO. FAC.
ENGENHARIA.

1.ASSUNTO I.COPPE/UFRJ II.TITULO(SERIE)

DEDICADO A ANA MARIA.

AGRADECIMENTOS:

AO AMIGO NELSON MACULAN,
PELO INCENTIVO CONSTANTE.

A PEDRO SALENBAUCH,
PELA ATENCAO E SUGESTOES.

A DEOCLECIANO PEGADO E FABIO C. FERREIRA,
PELA CONFIANCA NOS RESULTADOS DESTE TRABALHO.

SINOPSE

A TESE APRESENTA A PROPOSTA DE UMA LINGUAGEM DE MEDIO NIVEL PARA O MINICOMPUTADOR HP-2100A, DESCREVE O COMPILADOR IMPLEMENTADO E DISCUTE OS RESULTADOS OBTIDOS APOS UM ANO E MEIO DE EXPERIENCIA DE USO DA LINGUAGEM, NA PROGRAMACAO DE SISTEMAS.

ABSTRACT

THE AUTHOR PROPOSES A MEDIUM-LEVEL LANGUAGE FOR THE HP-2100A MINICOMPUTER, DESCRIBES THE STRUCTURE OF THE IMPLEMENTED COMPILER AND PRESENTS THE RESULTS OF ONE YEAR AND HALF OF SYSTEMS PROGRAMMING EXPERIENCE WITH THAT LANGUAGE.

INDICE DOS CAPITULOS

INDICE DOS APENDICES

I. INTRODUCAO

A LCS (LINGUAGEM PARA CONSTRUCAO DE SISTEMAS), DESENVOLVIDA JUNTO AO SERPRO (DIVISAO DE FABRICACAO) E VOLTADA PARA O HP-2100A, DEVEU-SE, COMO TODA LINGUAGEM DE NIVEL MEDIO E ADEQUADA APENAS A UMA CERTA MAQUINA, A NECESSIDADE, POR UM LADO, DE UM INSTRUMENTO DE PROGRAMACAO MAIS EFICIENTE E CONFIAVEL QUE A LINGUAGEM SIMBOLICA E, POR OUTRO, A CONSIDERACOES DE EXTENSAO DO CODIGO-OBJETO, FACE A MEMORIA DISPONIVEL.

AS APLICACOES A QUE SE DESTINA O HP-2100A NO SERPRO, BASICAMENTE O DESENVOLVIMENTO DE SISTEMAS DE TEMPO REAL PARA CONTROLE DA TRANSCRICAO DE DOCUMENTOS, NAO PERMITEM O USO DAS LINGUAGENS DE ALTO NIVEL PARA AS QUAIS EXISTEM COMPILADORES FORNECIDOS PELO FABRICANTE (FORTRAN E ALGOL), JA QUE OS PROGRAMAS, VIA DE REGRA BEM COMPLEXOS, NAO CABERIAM NA MEMORIA. A UNICA SOLUCAO VIAVEL ERA, DE INICIO, O "ASSEMBLER".

ENTRETANTO, A CODIFICACAO DE PROGRAMAS LONGOS EM LINGUAGEM SIMBOLICA APRESENTAVA SERIOS PROBLEMAS:

A) O TEXTO DE TAIS PROGRAMAS, DISPERSIVO E DE DIFICIL LEITURA, TORNAVA-OS POUCO COMPREENSIVEIS, MESMO PARA O PROPRIO AUTOR, APOS ALGUM TEMPO.

B) O GRAU DE LIBERDADE PERMITIDO AO PROGRAMADOR PELO "ASSEMBLER", TENDO COMO CONSEQUENCIA A MULTIPLICACAO DAS POSSIBILIDADES DE ERRO, TORNAVA OS PROGRAMAS POUCO CONFIAVEIS E MUITO SUJEITOS A PEQUENOS EQUIVOCOS.

C) A INEXISTENCIA DOS CONCEITOS DE ENTIDADES LOCAIS E GLOBAIS, ACARRETANDO A FUSAO DAS PARTES DE UM PROGRAMA EM UM TODO MAIS OU MENOS DISFORME, NO QUAL DE QUALQUER PONTO E' POSSIVEL A REFERENCIA A QUALQUER OUTRO PONTO, PREJUDICAVA BASTANTE A APLICACAO DAS TECNICAS DE PROGRAMACAO ESTRUTURADA. O CUSTO DA DEPURACAO E MODIFICACAO DE PROGRAMAS ERA, POR ESSE MOTIVO, MUITO ALTO.

O QUE SE BUSCOU, PORTANTO, COM A LCS, FOI A SUPERACAO DAS DIFICULDADES DA CODIFICACAO EM LINGUAGEM SIMBOLICA, ACIMA EXPOSTAS, E AGRAVADAS PELA AUSENCIA DE UM EXPANSOR DE MACRO-INSTRUCOES, SENDO POREM NECESSARIO EVITAR O INCONVENIENTE DAS LINGUAGENS DE ALTO NIVEL, DA PRODUCAO DE UM CODIGO-OBJETO DEMASIADO EXTENSO.

PARA A DEFINICAO DA LINGUAGEM, SERVIU DE FONTE INSPIRADORA O ARTIGO DE NIKLAUS WIRTH, PUBLICADO EM JANEIRO DE 1966 NO "JOURNAL OF THE A.C.M.", PROPONDO UMA LINGUAGEM DE MEDIO NIVEL

PARA O IBM-360: PL-360 (*1). OUTRAS REFERENCIAS FORAM: O MANUAL DE LP-15 (PARA O MITRA-15 DA CII), O MANUAL DE SPL-3000 (PARA O HP-3000) E O ARTIGO SOBRE A LINGUAGEM BLISS (PARA O PDP-10), PUBLICADO NA "COMMUNICATIONS OF THE A.C.M." (*3).

COMPARADA AO "ASSEMBLER", PODERIAMOS RELACIONAR AS SEGUINTES VANTAGENS, EM FAVOR DA LCS:

A) FORMA SINTATICA SEMELHANTE A DO ALGOL, COM A ADOCAO DE CONCEITOS COMO OS DE BLOCO, PROCEDIMENTO, EXPRESSAO, INSTRUCOES CONDICIONAIS (IF), ITERATIVAS (FOR, WHILE, REPEAT), E DE CHAVEAMENTO INDEXADO (CASE), ALEM DA NOCAO DE TIPO. OS PROGRAMAS TORNAM-SE MAIS LEGIVEIS E AUTO-DOCUMENTADOS, EXPRESSANDO DE FORMA BEM MAIS CLARA AS IDEIAS ENVOLVIDAS.

B) REGRAS SIMPLES E LOGICAS, DE FACIL APRENDIZADO; USO SEM MISTERIOS, REQUERENDO APENAS OS CONHECIMENTOS BASICOS DA ARQUITETURA DA MAQUINA.

C) ACESSO AOS RECURSOS ESSENCIAIS DO COMPUTADOR ("HARDWARE" E "SOFTWARE"), INCLUIDOS OS NECESSARIOS A CONSTRUCAO DE COMPILADORES E SISTEMAS OPERACIONAIS.

D) CODIGO-OBJETO SATISFATORIAMENTE OTIMIZADO, JA QUE O PROGRAMADOR CONTROLA AS OPERACOES A NIVEL DE REGISTRADORES. (MEDIDAS EFETUADAS INDICARAM QUE O USO DA LCS NAO ACARRETAVA, POR SI, UM AUMENTO DE MAIS DE 20% EM RELACAO AO "ASSEMBLER").

E) PROTECAO CONTRA OS ERROS DE EXECUCAO MAIS FREQUENTES, QUAIS SEJAM, OS DE INDEXACAO E ENDERECAMENTO.

AO DEFINIR-SE A LINGUAGEM TEVE-SE AINDA EM VISTA A CONSTRUCAO DE UM COMPILADOR DE UM SO PASSO. A PRIMEIRA VERSAO DO COMPILADOR, ESCRITA PARTE EM ALGOL PARTE EM "ASSEMBLER", REPRESENTA A IMPLEMENTACAO DA LINGUAGEM CONFORME DESCRITA A SEGUIR. ESTA' PREVISTA A FEITURA DE UMA SEGUNDA VERSAO, ENCAMINHANDO ALGUMAS EXTENSOES E A SER TOTALMENTE ESCRITA COM OS RECURSOS JA IMPLEMENTADOS.

O COMPILADOR FOI CONCLUIDO EM QUATRO MESES DE TRABALHO, FICANDO PRONTO EM JULHO DE 1975. DESDE ENTAO VEM SENDO CONSTANTEMENTE UTILIZADO, DESTACANDO-SE ENTRE AS APLICACOES DESENVOLVIDAS NA LINGUAGEM ATE AGORA (NOVEMBRO DE 1976):

A) UM SISTEMA DE JOGOS EM TEMPO REAL, PARA DEMONSTRACAO DOS TERMINAIS DE VIDEO PRODUZIDOS NO SERPRO.

B) AS PARTES MAIS COMPLEXAS DO SISTEMA DE TRANSCRICAO POR TERMINAIS DE VIDEO, STV-1600, APRESENTADO EM OUTUBRO DE 76 NO IX CONGRESSO NACIONAL DE PROCESSAMENTO DE DADOS; NOTADAMENTE, O INTERPRETADOR, O REFORMATADOR, O COMPILADOR, OS PROGRAMAS AUXILIARES DE RECUPERACAO DE ARQUIVOS E DE CONFIGURACAO DO SISTEMA.

COM BASE NA EXPERIENCIA ACUMULADA NESSE PERIODO, PODEMOS DIZER QUE A LINGUAGEM ATINGIU PLENAMENTE OS OBJETIVOS, PROPICIANDO ESPECIALMENTE UMA CONSIDERAVEL REDUCAO DO TEMPO DE ESCRITA E DEPURACAO DOS PROGRAMAS.

II. DESCRICAO RESUMIDA DO HP-2100A

1. DADOS BASICOS

A) "HARDWARE"

O HEWLETT-PACKARD 2100A E' UM MINICOMPUTADOR DE ATE 32K PALAVRAS DE 16 BITS. A CONFIGURACAO BASICA INCLUI VERIFICACAO DE PARIDADE, PROTECAO DE MEMORIA PARA SISTEMAS OPERACIONAIS E INTERRUPCAO POR QUEDA DE FORCA, COM REINICIO AUTOMATICO. AS CONFIGURACOES OPCIONAIS INCLUEM DOIS CANAIS DE ACESSO DIRETO A MEMORIA (DMA), ENTRADA E SAIDA MULTIPLEXADA, PAINEL DE CONTROLE E INTERFACES DE ENTRADA E SAIDA. O TEMPO DE ACESSO A MEMORIA E' DE 980 NANO-SEGUNDOS.

B) "SOFTWARE"

OS RECURSOS DE "SOFTWARE" INCLUEM, ALEM DO "ASSEMBLER", QUATRO LINGUAGENS DE ALTO NIVEL: FORTRAN II, FORTRAN IV, ALGOL E BASIC. A BIBLIOTECA DE PROGRAMAS UTILITARIOS INCLUI ROTINAS PARA APLICACOES MATEMATICAS, PARA TRACADO DE GRAFICOS, PARA MANUTENCAO DE PROGRAMAS-FONTE EM DISCO E PARA DEPURACAO DE PROGRAMAS. O USUARIO DISPOE AINDA DE DIVERSOS SISTEMAS OPERACIONAIS: BCS (BASIC CONTROL SYSTEM), MTS (MAGNETIC TAPE SYSTEM), DOS (DISC OPERATING SYSTEM), RTE (REAL TIME EXECUTIVE) E UM SISTEMA DE TEMPO COMPARTILHADO PARA PROGRAMACAO EM BASIC.

2. REGISTRADORES DE TRABALHO

OS REGISTRADORES DE TRABALHO DO HP-2100A, EXPLICITAMENTE REFERENCIAVEIS PELAS INSTRUCOES DE MAQUINA, SAO CINCO, SENDO TRES DE 16 BITS (A, B, S) E DOIS DE UM BIT (E, O).

A) REGISTRADOR A: DESEMPENHA O PAPEL DE UM ACUMULADOR, NO QUAL TRANSCORREM OPERACOES ARITMETICAS, LOGICAS OU DE DESLOCAMENTO, DETERMINADAS PELO PROGRAMA EM EXECUCAO; COINCIDE COM A PALAVRA DA MEMORIA DE ENDERECO 00000.

B) REGISTRADOR B: DESEMPENHA O PAPEL DE UM SEGUNDO ACUMULADOR, SEMELHANTE AO A, EMBORA NAO ADMITINDO OPERACOES LOGICAS NEM DE MULTIPLICACAO; COINCIDE COM A POSICAO DE MEMORIA DE ENDERECO 00001.

C) REGISTRADOR S: ENDERECAVEL COMO ELEMENTO PERIFERICO (CANAL 01), RECEBENDO E PASSANDO DADOS PARA OS REGISTRADORES A E B; SEU VALOR PODE SER MODIFICADO PELO REGISTRADOR EXISTENTE NO PAINEL DE OPERACAO.

D) BIT DE EXTENSAO (E): DESTINADO A ASSINALAR O VAI-UM EM OPERACOES DE SOMA.

E) BIT DE "OVERFLOW" (O): DESTINADO A ACUSAR ERRO EM OPERACOES DE SOMA OU DIVISAO.

3. INSTRUCOES DE MAQUINA

PODEM-SE GRUPAR EM CINCO CATEGORIAS, APRESENTADAS A SEGUIR:

3.1) INSTRUCOES DE REFERENCIA A MEMORIA

SAO INSTRUCOES DE CODIGO DE OPERACAO E ENDERECO REPRESENTADOS EM 16 BITS, QUE PERFAZEM AS OPERACOES ELEMENTARES ENVOLVENDO REFERENCIA A POSICOES DE MEMORIA.

A) LDA/LDB (LOAD A/B)
O VALOR DO OPERANDO E' CARREGADO NO REGISTRADOR A (LDA) OU B (LDB).

B) STA/STB (STORE A/B)
O VALOR CONTIDO NO REGISTRADOR A (STA) OU B (STB) E' ARMAZENADO NA POSICAO ENDERECADA.

C) ADA/ADB (ADD TO A/B)
E' SOMADO AO REGISTRADOR A (ADA) OU B (ADB) O VALOR DO OPERANDO.

D) AND/IOR/XOR (AND/INCLUSIVE OR/EXCLUSIVE OR)
E' EFETUADA SOBRE O REGISTRADOR A A OPERACAO LOGICA INDICADA, ENTRE O VALOR DESTE E O DO OPERANDO.

E) ISZ (INCREMENT AND SKIP IF ZERO)
O VALOR DO OPERANDO E' INCREMENTADO DE UMA UNIDADE; CASO RESULTE O VALOR ZERO, E' SALTADA A INSTRUCAO (SUPOSTAMENTE) CONTIDA NA PALAVRA SEGUINTE, DA MEMORIA.

F) JMP (JUMP)
DESVIA PARA A POSICAO ENDERECADA.

G) JSB (JUMP TO SUBROUTINE)
O ENDERECO DA POSICAO SEGUINTE, NA MEMORIA, E' ARMAZENADO NA POSICAO ENDERACADA, SENDO A EXECUCAO DESVIADA PARA A POSICAO SEGUINTE A ESSA. (O RETORNO A SEQUENCIA ANTERIOR PODE, PORTANTO, SER OBTIDO POR UM "JUMP" INDIRETO A POSICAO ENDERECADA PELO JSB).

3.2) INSTRUCOES DE REFERENCIA A REGISTRADOR

EM TODOS OS CASOS, INSTRUCOES DE 16 BITS.

A) NOP (NO OPERATION)

B) ALS/BLS (A/B LEFT SHIFT)
ARS/BRS (A/B RIGHT SHIFT)
DESLOCAMENTO ARITMETICO DE A OU B DE UMA POSICAO (UM BIT), PARA A ESQUERDA OU PARA A DIREITA.

C) RAL/RBL (ROTATE A/B LEFT)
RAR/RBR (ROTATE A/B RIGHT)
ROTACAO DO REGISTRADOR A OU B DE UMA POSICAO, PARA A ESQUERDA OU PARA A DIREITA.

D) ALR/BLR (A/B LEFT SHIFT, CLEAR SIGN)
SEMELHANTE A ALS/BLS, POREM SENDO ZERADO O BIT DE SINAL APOS O DESLOCAMENTO.

E) ELA/ELB (ROTATE E LEFT WITH A/B)
ERA/ERB (ROTATE E RIGHT WITH A/B)
ROTACAO DO REGISTRADOR A OU B, DE UMA POSICAO PARA A ESQUERDA OU PARA A DIREITA, EM CONJUNTO COM O BIT DE EXTENSAO.

F) ALF/BLF (ROTATE A/B LEFT FOUR)
EQUIVALENTE A QUATRO INSTRUCOES RAL/RBL SUCESSIVAS.

G) CLA/CLB (CLEAR A/B)
CMA/CMB (COMPLEMENT A/B)
CCA/CCB (CLEAR AND COMPLEMENT A/B)
INSTRUCOES PARA ZERAR, COMPLEMENTAR OU LIGAR TODOS OS BITS DE A OU B.

H) INA/INB (INCREMENT A/B)
INSTRUCOES PARA INCREMENTAR A OU B.

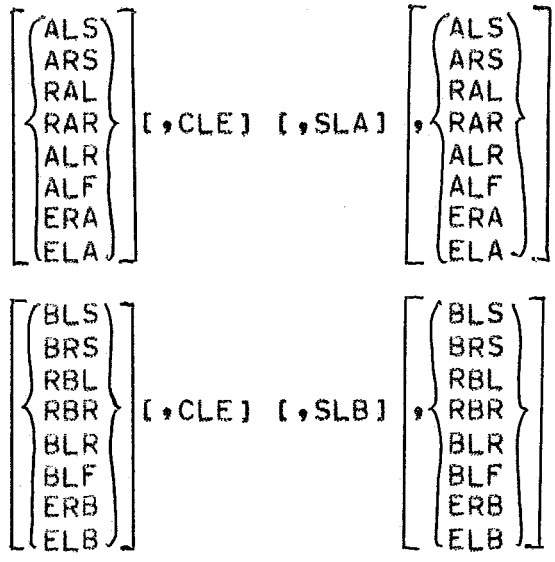
I) CLE (CLEAR E)
CME (COMPLEMENT E)
CCE (CLEAR AND COMPLEMENT E)
INSTRUCOES PARA ZERAR, COMPLEMENTAR OU LIGAR O BIT DE EXTENSAO.

J) SEZ (SKIP IF E IS ZERO)
SSA/SSB (SKIP IF SIGN OF A/B IS ZERO)
SLA/SLB (SKIP IF LEAST SIGNIFICANT BIT OF A/B IS ZERO)
SZA/SZB (SKIP IF A/B IS ZERO)
INSTRUCOES PARA SALTAR CONDICIONALMENTE UMA PALAVRA DO CODIGO-OBJETO.

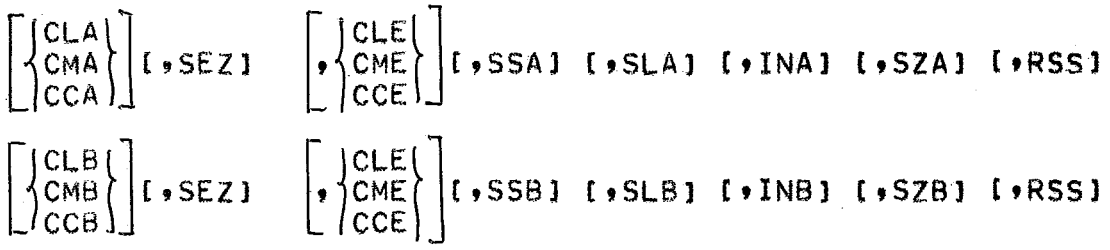
K) RSS (REVERSE SKIP SENSE)
INSTRUCAO PARA O SALTO INCONDICIONAL DE UMA PALAVRA DO CODIGO-OBJETO.

DETERMINADAS INSTRUCOES DE REFERENCIA A REGISTRADOR PODEM SER COMBINADAS EM UMA UNICA INSTRUCAO DE MAQUINA, CONFORME OS QUADROS ABAIXO:

QUADRO I



QUADRO II



VALEM AS SEGUINTES REGRAS: SOMENTE UMA INSTRUCAO PODE SER ESCOLHIDA DE UMA MESMA COLUNA; REFERENCIAS AOS REGISTRADORES A E B NAO PODEM SER MISTURADAS; A SEQUENCIA DE EXECUCAO E' A DA ESQUERDA PARA A DIREITA.

3.3) INSTRUCOES DE ENTRADA E SAIDA

SAO INSTRUCOES DE 16 BITS, COMPOSTOS BASICAMENTE DE CODIGO DE OPERACAO E CANAL REFERENCIADO.

A) HLT (HALT)
COLOCA O PROCESSADOR EM TEMPO DE PARADA ("HALT MODE").

B) LIA/LIB (LOAD INTO A/B)

MIA/MIB (MERGE WITH A/B)
OPERACOES DE ENTRADA DE DADOS (VIA REGISTRADOR).

C) OTA/OTB (OUTPUT A/B)
OPERACOES DE SAIDA DE DADOS (VIA REGISTRADOR).

D) STF/CLF (SET/CLEAR FLAG)
STC/CLC (SET/CLEAR CONTROL)
OPERACOES PARA LIGAR OU DESLIGAR O BIT DE SINALIZACAO ("FLAG") E O BIT DE CONTROLE ("CONTROL") DO CANAL ESPECIFICADO.

E) STO/CLO (SET/CLEAR OVERFLOW)
OPERACOES PARA LIGAR OU DESLIGAR O BIT DE "OVERFLOW".

F) SFS/SFC (SKIP IF FLAG SET/CLEAR)
OPERACOES PARA SALTAR A PROXIMA PALAVRA DO CODIGO-OBJETO, CONFORME O ESTADO DO BIT DE SINALIZACAO DO CANAL ESPECIFICADO.

G) SOS/SOC (SKIP IF OVERFLOW SET/CLEAR)
OPERACOES PARA SALTAR A PROXIMA PALAVRA DO CODIGO-OBJETO, CONFORME O ESTADO DO BIT DE "OVERFLOW".

3.4) INSTRUCOES ADICIONAIS DE REFERENCIA A MEMORIA

SAO INSTRUCOES PROCESSADAS PELA UNIDADE ARITMETICA ESTENDIDA (EAU), DISPOSITIVO OPCIONAL DO HP-2100A, POREM QUASE SEMPRE ADOTADO. DIFEREM DAS INSTRUCOES DE REFERENCIA A MEMORIA DESCRITAS NO ITEM 3.1 POR OCUPAREM DUAS PALAVRAS, A PRIMEIRA DAS QUAIS REPRESENTANDO O CODIGO DE OPERACAO (INCLUSIVE O BIT DE ENDERECAMENTO INDIRETO) E A SEGUNDA O ENDERECO DO OPERANDO.

A) MPY (MULTIPLY)
MULTIPLICA O CONTEUDO DO REGISTRADOR A PELO VALOR DO OPERANDO; RESULTA UM INTEIRO DE 32 BITS, REPRESENTADO PELA CONCATENACAO DE B E A (NESTA ORDEM).

B) DIV (DIVIDE)
DIVIDE PELO VALOR DO OPERANDO O INTEIRO DE 32 BITS REPRESENTADO EM B E A; RESULTA O QUOCIENTE EM A E O RESTO EM B.

C) DLD (DOUBLE LOAD)
CARREGA EM A O CONTEUDO DA POSICAO ENDERECADA E EM B O DA POSICAO SEGUINTE.

3.5) INSTRUCOES ADICIONAIS DE REFERENCIA A REGISTRADOR

PROCESSADAS TAMBEM APENAS PELA UNIDADE ARITMETICA ESTENDIDA (EAU). OCUPAM UMA PALAVRA E TEM POR FUNCAO O DESLOCAMENTO OU ROTACAO EM CONJUNTO DOS REGISTRADORES B E A, TOMADOS COMO UM REGISTRADOR DE 32 BITS. O NUMERO DE POSICOES DESLOCADAS OU

RODADAS PODE VARIAR DE 1 A 16 E E' REPRESENTADO NOS 4 ULTIMOS BITS DA INSTRUCAO.

A) ASL/ASR (ARITHMETIC SHIFT LEFT/RIGHT)
DESLOCAMENTO DE B E A EM CONJUNTO, PARA A ESQUERDA OU PARA A DIREITA.

B) LSL/LSR (LOGICAL SHIFT LEFT/RIGHT)
DESLOCAMENTO LOGICO DE B E A EM CONJUNTO, PARA A ESQUERDA OU PARA A DIREITA.

C) RRL/RRR (ROTATE LEFT/RIGHT)
ROTACAO DE B E A EM CONJUNTO, PARA A ESQUERDA OU PARA A DIREITA.

4. ENDERECAMENTO

A MEMORIA DO HP-2100A E' FORMADA DE UM NUMERO INTEIRO DE PAGINAS DE 1024 PALAVRAS.

A PAGINA E' DEFINIDA COMO A MAIOR PORCAO DE MEMORIA DIRETAMENTE ENDERECAVEL PELOS BITS QUE DESIGNAM O OPERANDO, NAS INSTRUCOES DE REFERENCIA A MEMORIA DESCRITAS NO ITEM 3.1. COMO SAO RESERVADOS OS 10 ULTIMOS BITS PARA ESSE FIM, EM UMA INSTRUCAO DAQUELE TIPO, O TAMANHO DA PAGINA E' FIXADO EM 1024 PALAVRAS. OS CINCO PRIMEIROS BITS INDICAM O CODIGO DE OPERACAO E O SEXTO BIT ESPECIFICA SE O OPERANDO ENDERECADO ESTA' NA PAGINA ZERO OU NA PAGINA CORRENTE.

PORTANTO, AS INSTRUCOES DE REFERENCIA A MEMORIA SO REFERENCIAM DIRETAMENTE A PAGINA ZERO OU A CORRENTE, SENDO O ACESSO A OUTRAS PAGINAS RESOLVIDO NECESSARIAMENTE POR ENDERECAMENTO INDIRETO.

AS REFERENCIAS INDIRETAS SAO PROCESSADAS EM CASCATA, DO SEGUINTE MODO:
SE O BIT 15 (MAIS A ESQUERDA) DE UMA INSTRUCAO DE REFERENCIA A MEMORIA ESTA' LIGADO, OS BITS DE ENDERECO INDICAM, NA PAGINA ZERO OU NA CORRENTE, A POSICAO ONDE SE ENCONTRA O ENDERECO DO OPERANDO. SE ESSE ENDERECO APRESENTAR, POR SUA VEZ, O BIT 15 ACESO, OS DEMAIS BITS INDICAM O ENDERECO (EM QUALQUER PAGINA) DO OPERANDO, E ASSIM SUCESSIVAMENTE, ATE QUE SE ENCONTRE UM ENDERECO DE BIT 15 DESLIGADO, O QUAL INDICARA A POSICAO DO OPERANDO DESEJADO.

A PAGINA ZERO E' EM GERAL QUASE QUE TOTALMENTE RESERVADA PARA O ESTABELECIMENTO DE LIGACOES ENTRE AS DIVERSAS PAGINAS DA MEMORIA.

5. IMPLICACOES DA ARQUITETURA DO HP-2100A

O CONJUNTO DE INSTRUCOES DO HP-2100A PODE SER CONSIDERADO BASTANTE RUDIMENTAR. NAO HA INDEXADORES NEM NADA SEMELHANTE: O ACESSO AO ELEMENTO DE UM VETOR SO E' POSSIVEL POR ENDERECAMENTO INDIRETO. NAO HA INSTRUCOES DE REFERENCIA A CAMPOS DE BITS, NEM MESMO A BYTES. NAO HA OPERACOES ARITMETICAS PARA INTEIROS DUPLOS NEM QUALQUER ESBOCO DE OPERACOES EM PILHA. AS INSTRUCOES DE ROTACAO E DESLOCAMENTO SAO DE UMA POBREZA SURPREENDENTE. PARA SE DESLOCAR O CONTEUDO DE UM REGISTRADOR DE N BITS, PARA A ESQUERDA OU PARA A DIREITA, SENDO N OBTIDO EM TEMPO DE EXECUCAO, E' NECESSARIO MODIFICAR O PROPRIO CODIGO EXECUTAVEL.

NESSE CONTEXTO, NAO FOI POSSIVEL MANTER PARA A LCS O PRINCIPIO, COMUM A MAIORIA DAS LINGUAGENS DE NIVEL MEDIO, DA NAO-EXISTENCIA DE UMA BIBLIOTECA DE ROTINAS INTRINSECAS, ISTO E', AQUELAS PARA AS QUAIS O PROPRIO COMPILADOR INTRODUZ CHAMADAS, NOS CASOS QUE NAO CONSEGUE RESOLVER COM ALGUMAS (POUCAS) INSTRUCOES DE MAQUINA. ESSES CASOS, PARA A LCS, SAO OS SEGUINTES:

A) TRANSMISSAO DE PARAMETROS;
B) INDEXACAO (VETORES DE TIPO CURTO E DUPLO);
C) CHAVEAMENTO INDEXADO (INSTRUCAO CASE);
D) COMPARACAO COM VARIAVEIS OU CONSTANTES DE TIPO CURTO (EXCETO PARA OPERADORES DE IGUALDADE E DESIGUALDADE E COMPARACOES COM ZERO);
E) COMPARACAO COM VARIAVEIS E CONSTANTES DE TIPO DUPLO (TODAS);
F) OPERACOES DE MULTIPLICACAO, DIVISAO E RESTO PARA VARIAVEIS DE TIPO CURTO (OPERADORES *,/,!);
G) OPERACOES DE SOMA, COMPLEMENTACAO ARITMETICA, MULTIPLICACAO, DIVISAO E RESTO PARA VARIAVEIS DE TIPO DUPLO;
H) DESLOCAMENTO E ROTACAO VARIAVEIS, PARA REGISTRADORES DE TIPOS CURTO E DUPLO.

A INTRODUCAO DE CHAMADAS DE ROTINA, A REVELIA DO PROGRAMADOR, EM UMA MAQUINA CUJA ARQUITETURA NAO PREVE A COMPOSICAO DE PROGRAMAS REENTRANTES, NAO SE DA SEM PROBLEMAS. DE FATO, SE UM PROCESSO X E' INTERROMPIDO EM MEIO A EXECUCAO DE UMA ROTINA INTRINSECA E SE O CONTROLE E' ENTAO TRANSFERIDO PARA UM PROCESSO Y QUE UTILIZE A MESMA ROTINA, O RETORNO A X REDUNDARA EM ERRO. ESSA SERIA A DIFICULDADE DE ESCREVER EM LCS ROTINAS PARA CAPTURA DE INTERRUPCOES.

A SOLUCAO PRIMEIRAMENTE IMAGINADA FOI A DE DESLIGAR O SISTEMA DE INTERRUPCOES DURANTE A EXECUCAO DE QUALQUER ROTINA INTRINSECA. HAVIA POREM O INCONVENIENTE DE SE ONERAR SEM VANTAGEM O PROCESSAMENTO DOS PROGRAMAS QUE NAO COMPARTILHASSEM ROTINAS COM OUTROS CAPAZES DE INTERROMPE-LOS. OPTOU-SE ENTAO PELA EXISTENCIA DE DUAS BIBLIOTECAS DE ROTINAS INTRINSECAS,

FICANDO A CARGO DO PROGRAMADOR INDICAR (POR MEIO DE UM CARTAO DE CONTROLE) O CASO EM QUE SE ENQUADRASSE O SEU PROGRAMA.

O QUE A EXPERIENCIA FINALMENTE APONTOU, ENTRETANTO, FOI A INUTILIDADE DA BIBLIOTECA CUJAS ROTINAS CHAVEASSEM AS INTERRUPCOES. ISTO PORQUE VALIA MAIS A PENA GASTAR ALGUM ESFORCO ESCREVENDO AS ROTINAS DE INTERRUPCAO EM LINGUAGEM SIMBOLICA, JA QUE ESSAS ROTINAS SAO EM GERAL BASTANTE SIMPLES, QUE ESCREVER TODO O SISTEMA EM LCS E TER COMO CONTRAPARTIDA UMA SOBRECARGA NO TEMPO DE EXECUCAO.

OUTRAS IMPLICACOES IMPORTANTES DA ARQUITETURA DO HP-2100A, NOTADAMENTE A AUSENCIA DE OPERACOES EM PILHA, SAO:

A) A NAO-IMPLEMENTACAO DA ALOCACAO DINAMICA DE MEMORIA;
B) A NAO-IMPLEMENTACAO DE PROCEDIMENTOS RECURSIVOS;
C) A REDUCAO DAS EXPRESSOES A UMA FORMA RESTRITA, SEM NIVEIS DE PARENTESES NEM PRECEDENCIA DE OPERADORES (MODELO PL-360).

EM CONTRAPARTIDA, O ESQUEMA DE ENDERECAMENTO INDIRETO EM CASCATA, A PARTIR DA ADOCAO DE UMA ROTINA INTRINSECA PARA A TRANSMISSAO DE PARAMETROS, PERMITIU A FACIL IMPLEMENTACAO DOS CONCEITOS DE PARAMETRO FORMAL E DE CHAMADAS "POR NOME" E "POR VALOR", O QUE A PRATICA DEMONSTROU DE GRANDE VALIA.

6. SOBRE A DESCRICAO DA LINGUAGEM

ANTES DE PASSARMOS AO PROXIMO CAPITULO, EM QUE E' DESCRITA A LINGUAGEM, GOSTARIAMOS DE ASSINALAR OS MOTIVOS DE TERMOS OPTADO POR PALAVRAS RESERVADAS EM INGLES.

A LCS E' UMA LINGUAGEM DESTINADA A UM COMPUTADOR DE FABRICACAO ESTRANGEIRA. TODA A LITERATURA QUE O ACOMPANHA E' ESCRITA EM INGLES E ISSO SE REFLETE NO VOCABULARIO DE SEUS USUARIOS POR UMA FORTE CARGA DE TERMOS NAQUELE IDIOMA. ORA, NESSE QUADRO, SERIA BASTANTE FORCADO (TALVEZ ATE PEDANTE) TRADUZIR PALAVRAS JA CONSAGRADAS PELO USO, COMO "FLAG" (NO SENTIDO AQUI ADOTADO PELO FABRICANTE), "OVERFLOW" E OUTRAS; OU, POR EXEMPLO, ARRANJAR SIGLAS EM PORTUGUES QUE EQUIVALESSEM AOS MNEMONICOS MIA ("MERGE WITH A") OU OTB ("OUTPUT B").

OPTAMOS PORTANTO PELA UNIFORMIZACAO EM INGLES, ACHANDO MAIS JUSTO BATALHAR PELA UTILIZACAO INTEGRAL DO PORTUGUES EM LINGUAGENS PARA SISTEMAS DE PROJETO NACIONAL.

III. DESCRICAO DA LINGUAGEM

1. LETRAS E ALGARISMOS

```
<LETRA> ::= A/B/C/D/E/F/G/H/I/J/K/L/M/N/
            O/P/Q/R/S/T/U/V/W/X/Y/Z/$
<ALGARISMO> ::= 0/1/2/3/4/5/6/7/8/9
```

AS LETRAS SERVEM PARA FORMAR OS IDENTIFICADORES E AS CADEIAS DE CARACTERES. OS ALGARISMOS SERVEM PARA FORMAR OS IDENTIFICADORES, AS CONSTANTES NUMERICAS E AS CADEIAS DE CARACTERES.

2. IDENTIFICADORES

```
<IDENTIFICADOR> ::= <LETRA> / <IDENTIFICADOR> <LETRA> /
      <IDENTIFICADOR> <ALGARISMO> /
      <IDENTIFICADOR>.
<IDENTIFICADOR DE VARIAVEL SIMPLES> ::= <IDENTIFICADOR>
<IDENTIFICADOR DE VETOR> ::= <IDENTIFICADOR>
<IDENTIFICADOR DE ROTULO> ::= <IDENTIFICADOR>
<IDENTIFICADOR DE PROCEDIMENTO> ::= <IDENTIFICADOR>
```

OS IDENTIFICADORES SERVEM PARA NOMEAR AS ENTIDADES DE QUE TRATA UM PROGRAMA NA LINGUAGEM: VARIAVEIS, ROTULOS E PROCEDIMENTOS.

DOIS IDENTIFICADORES SO SAO CONSIDERADOS DISTINTOS PELO COMPILADOR SE DIFERIREM EM PELO MENOS UM DE SEUS 15 PRIMEIROS CARACTERES.

3. CONSTANTES NUMERICAS

```
<CONSTANTE NUMERICA> ::= <CONSTANTE CURTA> /
      <CONSTANTE DUPLA>
<CONSTANTE CURTA>  ::= <CONSTANTE CURTA SEM SINAL> /
      <SINAL> <CONSTANTE CURTA SEM SINAL>
<CONSTANTE CURTA SEM SINAL> ::= <INTEIRO SEM SINAL> /
      <OCTAL>
<CONSTANTE DUPLA> ::= <CONSTANTE DUPLA SEM SINAL> /
      <SINAL> <CONSTANTE DUPLA SEM SINAL>
<CONSTANTE DUPLA SEM SINAL> ::= <INTEIRO SEM SINAL>D
<INTEIRO SEM SINAL> ::= <ALGARISMO> /
      <INTEIRO SEM SINAL> <ALGARISMO>
<OCTAL> ::= <SEQUENCIA OCTAL>B
<SEQUENCIA OCTAL> ::= <ALGARISMO OCTAL> /
      <SEQUENCIA OCTAL> <ALGARISMO OCTAL>
<ALGARISMO OCTAL> ::= 0/1/2/3/4/5/6/7
```

```
<SINAL> ::= + / -
```

AS CONSTANTES NUMERICAS DITAS DE TIPO CURTO SAO AS REPRESENTADAS EM 16 BITS; AS DUPLAS, EM 32 BITS.

4. CADEIAS DE CARACTERES

```
<CADEIA DE CARACTERES> ::= "<SEQUENCIA DE CARACTERES>"
<SEQUENCIA DE CARACTERES> ::= <CARATER> /
      <SEQUENCIA DE CARACTERES> <CARATER>
<CARATER> ::= <LETRA> / <ALGARISMO> / <BARRA> /
      <CARATER ESPECIAL>
<BARRA> ::= /
<CARATER ESPECIAL> ::=  /!/#/%/&/'/(/)/*/+/,/-/./
      :/;/</=/>/?/@/[/]/^/""
```

A CADEIA DE UM SO CARATER E' ARMAZENADA NO BYTE MAIS A DIREITA DE UMA PALAVRA, SENDO O BYTE MAIS A ESQUERDA ZERADO. AS DEMAIS CADEIAS SAO REPRESENTADAS EM DOIS CARACTERES POR PALAVRA, SENDO AS DE COMPRIMENTO IMPAR COMPLETADAS POR UM BRANCO. O CARATER " (ASPA DUPLA) E' DENOTADO "" (DUAS ASPAS DUPLAS CONSECUTIVAS). O COMPILADOR NAO PERMITE CADEIAS DE MAIS DE 128 CARACTERES.

5. REGISTRADORES, CONTROLES, FLAGS E BITS ESPECIAIS

```
<REGISTRADOR> ::= <REGISTRADOR CURTO> /
     <REGISTRADOR DUPLO>
<REGISTRADOR CURTO> ::= RA / RB
<REGISTRADOR DUPLO> ::= RBA
<CONTROLE> ::= CONTROL( <CANAL> )
<FLAG> ::= FLAG( <CANAL> )
<BIT ESPECIAL> ::= RA0 / RB0 / EXT / OVFL
<CANAL> ::= <CONSTANTE NUMERICA>
```

NOTA: RA0 E RB0 REPRESENTAM, RESPECTIVAMENTE, OS BITS DE MAIS BAIXA ORDEM DE RA E RB.

6. TIPOS

```
<TIPO> ::= <TIPO SIMPLES> / <TIPO MULTIPLO>
<TIPO SIMPLES> ::= <TIPO CURTO> / <TIPO DUPLO>
<TIPO CURTO> ::= WORD
<TIPO DUPLO> ::= DOUBLE
<TIPO MULTIPLO> ::= <VETOR CURTO> / <VETOR DUPLO>
<VETOR CURTO> ::= WORD( <DIMENSAO> )
<VETOR DUPLO> ::= DOUBLE( <DIMENSAO> )
<DIMENSAO> ::= <CONSTANTE NUMERICA>
```

O TIPO SIMPLES REFERE-SE A UMA CELULA (DE UMA OU DUAS PALAVRAS), NOMEADA INDIVIDUALMENTE. O TIPO MULTIPLO REFERE-SE A UMA SEQUENCIA DE CELULAS, A CADA UMA DAS QUAIS E' ATRIBUIDO UM MESMO TIPO SIMPLES.

7. VARIAVEIS

<VARIAVEL> ::= <VARIAVEL SIMPLES> /
<VARIAVEL INDEXADA>
<VARIAVEL SIMPLES> ::= <IDENTIFICADOR DE VARIAVEL SIMPLES>
<VARIAVEL INDEXADA> ::= <IDENTIFICADOR DE VETOR>(<INDICE>)
<INDICE> ::= <VARIAVEL SIMPLES> / <REGISTRADOR> /
<CONSTANTE NUMERICA>

VARIAVEL E' A DESIGNACAO DO VALOR DE UMA DETERMINADA CELULA DA MEMORIA.
(NA VARIAVEL INDEXADA, O INDICE DEVE SER DE TIPO CURTO; A PRIMEIRA CELULA DE UM VETOR E' A DE INDICE NULO.)

8. DECLARACOES
<DECLARACAO> ::= <DECLARACAO DE VARIAVEL> /
<DECLARACAO DE ROTULO> /
<DECLARACAO DE PROCEDIMENTO>

AS DECLARACOES INTRODUZEM ENTIDADES A SEREM TRATADAS PELO PROGRAMA.

9. DECLARACAO DE VARIAVEL

<DECLARACAO DE VARIAVEL> ::= <DECLARACAO POR ALOCACAO> /
<DECLARACAO POR SUPERPOSICAO>
<DECLARACAO POR ALOCACAO> ::= <TIPO> <ITEM DE ALOCACAO> /
<DECLARACAO POR ALOCACAO>, <ITEM DE ALOCACAO>
<ITEM DE ALOCACAO> ::= <IDENTIFICADOR> /
<IDENTIFICADOR> = <VALOR INICIAL> /
<IDENTIFICADOR> = (<LISTA DE VALORES INICIAIS>)
<VALOR INICIAL> ::= <VALOR INICIAL SIMPLES> /
<FATOR DE REPETICAO> : <VALOR INICIAL SIMPLES>
<FATOR DE REPETICAO> ::= <CONSTANTE NUMERICA>
<VALOR INICIAL SIMPLES> ::= <CONSTANTE NUMERICA> /
<CADEIA DE CARACTERES>
<LISTA DE VALORES INICIAIS> ::= <VALOR INICIAL> /
<LISTA DE VALORES INICIAIS>, <VALOR INICIAL>
<DECLARACAO POR SUPERPOSICAO> ::=
<TIPO> <ITEM DE SUPERPOSICAO> /
<DECLARACAO POR SUPERPOSICAO>, <ITEM DE SUPERPOSICAO>
<ITEM DE SUPERPOSICAO> ::= <IDENTIFICADOR> DEF <POSICAO>
<POSICAO> ::= <CONSTANTE NUMERICA> /
<IDENTIFICADOR DE VETOR> /

<IDENTIFICADOR DE VETOR> (<CONSTANTE NUMERICA>)

A DECLARACAO POR ALOCACAO DETERMINA A ALOCACAO DE UMA CERTA AREA PARA A VARIAVEL, A QUAL PODE SER INICIALIZADA EM TEMPO DE COMPILACAO.

A DECLARACAO POR SUPERPOSICAO PERMITE SUPERPOR UMA VARIAVEL A OUTRA, ANTERIORMENTE DECLARADA, OU ATRIBUIR A UMA VARIAVEL UM ENDERECO ABSOLUTO.

EXEMPLOS:

```
WORD ALFA,GAMA
WORD JOTA=1745B,PTO=".",TEMP
DOUBLE CONTADOR=0D
WORD(5) VET1=(5:0), VET2=(2:-1,0,2:+1)
WORD(40) MENSAGEM=("MENSAGEM",36:" ")
WORD(35) TEXTO DEF MENSAGEM(6)
WORD(2000B) PAGBASE DEF 00000B
```

OBSERVACOES:

A) SE VETX E' UM VETOR (CURTO OU DUPLO) DE DIMENSAO N, O SEU ELEMENTO DE MAIS BAIXA ORDEM E' VETX(0) E O DE MAIS ALTA ORDEM E'VETX(N-1).

B) O VALOR INICIAL DE UMA VARIAVEL DEVE SER COMPATIVEL COM O SEU TIPO. SAO COMPATIVEIS COM O TIPO CURTO: AS CONSTANTES CURTAS E AS CADEIAS DE CARACTERES; COM O TIPO DUPLO, APENAS AS CONSTANTES DUPLAS.

10. DECLARACAO DE ROTULO

```
<DECLARACAO DE ROTULO> ::= LABEL <IDENTIFICADOR> /
     <DECLARACAO DE ROTULO>, <IDENTIFICADOR>
```

ROTULOS INDICAM POSICOES DO PROGRAMA PARA AS QUAIS PODEM OCORRER DESVIOS.

A CADA ROTULO DECLARADO EM UM BLOCO DEVE CORRESPONDER, NO CORPO DO MESMO BLOCO, UMA DEFINICAO DE ROTULO (VER ADIANTE).

EXEMPLOS:

```
LABEL ROTA
LABEL $1,$2,$3
```

11. DECLARACAO DE PROCEDIMENTO

```
<DECLARACAO DE PROCEDIMENTO> ::= <PROCEDIMENTO INTERNO> /
     <PROCEDIMENTO EXTERNO>
<PROCEDIMENTO INTERNO> ::= <CABECALHO>; <INSTRUCAO>
<PROCEDIMENTO EXTERNO> ::= <CABECALHO>; CODE
<CABECALHO> ::= PROCEDURE <IDENTIFICADOR> /
     PROCEDURE <IDENTIFICADOR> (
```

```
                    <ESPECIFICACAO DE PARAMETROS> )
<ESPECIFICACAO DE PARAMETROS> ::= <ESPECIFICACAO> /
     <ESPECIFICACAO DE PARAMETROS>, <ESPECIFICACAO>
<ESPECIFICACAO> ::= <ESPECIFICADOR> <IDENTIFICADOR> /
     <ESPECIFICACAO>, <IDENTIFICADOR>
<ESPECIFICADOR> ::= <TIPO SIMPLES> /
     <TIPO SIMPLES> VALUE /
     <TIPO SIMPLES> (*) /
     LABEL / PROCEDURE
```

A DECLARACAO DE PROCEDIMENTO ASSOCIA UM IDENTIFICADOR A UMA INSTRUCAO, A QUAL SERA ATIVADA POR UMA INSTRUCAO DE CHAMADA (VER ADIANTE). OS PROCEDIMENTOS DITOS EXTERNOS SAO OS COMPILADOS SEPARADAMENTE.

12. INSTRUCOES

```
<INSTRUCAO> ::= <INSTRUCAO VAZIA> /
     <BLOCO> /
     <INSTRUCAO DE CHAMADA> /
     <INSTRUCAO DE BIT> /
     <INSTRUCAO DE REGISTRADOR> /
     <INSTRUCAO DE ARMAZENAMENTO> /
     <INSTRUCAO DE INCREMENTACAO> /
     <INSTRUCAO DE DESVIO> /
     <INSTRUCAO CONDICIONAL> /
     <INSTRUCAO DE CHAVEAMENTO> /
     <INSTRUCAO ITERATIVA> /
     <INSTRUCAO DE E/S>
```

13. INSTRUCAO VAZIA

```
<INSTRUCAO VAZIA> ::=
```

A INSTRUCAO VAZIA E' DENOTADA PELA AUSENCIA DE SIMBOLOS.

14. BLOCO

```
<BLOCO> ::= <CORPO DO BLOCO> <INSTRUCAO> END
<CORPO DO BLOCO> ::= <INICIO DO BLOCO> /
     <CORPO DO BLOCO> <INSTRUCAO>; /
     <CORPO DO BLOCO> <DEFINICAO DE ROTULO>
<INICIO DO BLOCO> ::= BEGIN /
     <INICIO DO BLOCO> <DECLARACAO>;
<DEFINICAO DE ROTULO> ::= <IDENTIFICADOR DE ROTULO>:
```

O BLOCO ATENDE A DUAS FINALIDADES:

A) REUNIR UMA SEQUENCIA DE INSTRUCOES EM UMA UNICA INSTRUCAO;
B) INTRODUZIR UM NOVO NIVEL DE NOMENCLATURA, PERMITINDO A

DECLARACAO DE NOVAS VARIAVEIS, PROCEDIMENTOS E ROTULOS.

NOTA: OS SIMBOLOS BEGIN E END PODEM SER RESPECTIVAMENTE DENOTADOS [E].

15. INSTRUCAO DE CHAMADA

```
<INSTRUCAO DE CHAMADA> ::= <IDENTIFICADOR DE PROCEDIMENTO> /
     <IDENTIFICADOR DE PROCEDIMENTO> (
               <LISTA DE PARAMETROS EFETIVOS> )
<LISTA DE PARAMETROS EFETIVOS> ::= <PARAMETRO EFETIVO> /
     <LISTA DE PARAMETROS EFETIVOS>, <PARAMETRO EFETIVO>
<PARAMETRO EFETIVO> ::= <CONSTANTE NUMERICA> /
     <CADEIA DE CARACTERES> /
     <VARIAVEL> /
     <REGISTRADOR> /
     <IDENTIFICADOR DE VETOR> /
     <IDENTIFICADOR DE ROTULO> /
     <IDENTIFICADOR DE PROCEDIMENTO>
```

A INSTRUCAO DE CHAMADA TEM POR EFEITO A ATIVACAO DE UM CERTO PROCEDIMENTO. FINDA A EXECUCAO DESTE, SERA EXECUTADA A INSTRUCAO QUE SE SEGUE A DE CHAMADA, A MENOS QUE O PROCEDIMENTO ATIVADO TERMINE POR UMA INSTRUCAO DE HALT OU DE DESVIO PARA FORA.

OS PARAMETROS EFETIVOS DEVEM CORRESPONDER EM TIPO AOS PARAMETROS FORMAIS, PELA ORDEM. PARAMETROS EFETIVOS CONSTANTES OU REGISTRADORES DEVEM CORRESPONDER A PARAMETROS FORMAIS ESPECIFICADOS POR VALOR ("BY VALUE"). PARAMETROS EFETIVOS DA FORMA <IDENTIFICADOR DE VETOR> DEVEM CORRESPONDER A PARAMETROS FORMAIS ESPECIFICADOS: <TIPO SIMPLES> (*) .

EXEMPLO:

SEJA O PROCEDIMENTO:

```
PROCEDURE PROCX(WORD VALUE X,Y, DOUBLE RES,
                WORD(*) AREA, LABEL ANORM);
     CODE
```

SEJAM AINDA AS DECLARACOES (OU ESPECIFICACOES):

```
WORD NK,P
DOUBLE RESULT
WORD(N) CARTAO,REG
LABEL ERRO
```

SAO VALIDAS AS SEGUINTES CHAMADAS:

```
PROCX(-1,P,RESULT,CARTAO,ERRO)
PROCX(REG(RA),RB,RESULT,CARTAO,ERRO)
```

OBSERVACOES:

A) UMA CADEIA DE CARACTERES, TRANSMITIDA COMO PARAMETRO, NAO DEVERA EXCEDER O COMPRIMENTO DE UMA PALAVRA.

B) O ESPECIFICADOR <TIPO SIMPLES> VALUE INDICA QUE, NA POSICAO CORRESPONDENTE AO PARAMETRO FORMAL, A ROTINA INTRINSECA QUE

RESOLVE A TRANSMISSAO DE PARAMETROS COLOCARA UMA COPIA DO VALOR DO PARAMETRO EFETIVO. NOS DEMAIS CASOS, E' TRANSMITIDO O ENDERECO DO PARAMETRO EFETIVO.

16. INSTRUCAO DE BIT

```
<INSTRUCAO DE BIT> ::= <FLAG> := <VALOR BINARIO> /
      <CONTROLE> := <VALOR BINARIO> /
      <BIT ESPECIAL> := <VALOR BINARIO>
<VALOR BINARIO> ::= 0 / 1
```

A INSTRUCAO DE BIT SERVE PARA LIGAR OU DESLIGAR "FLAGS", CONTROLES E BITS ESPECIAIS.

```
EXEMPLOS:
FLAG(0):=0
OVFL:=1
```

17. INSTRUCAO DE REGISTRADOR

```
<INSTRUCAO DE REGISTRADOR> ::= <REGISTRADOR> := <EXPRESSAO>
<EXPRESSAO> ::= <TERMO> /
      <ENDERECO> /
      <OPERADOR UNARIO> <TERMO> /
      <OPERADOR UNARIO> <ENDERECO> /
      <EXPRESSAO> <OPERADOR BINARIO> <TERMO>
<TERMO> ::= <CONSTANTE NUMERICA> /
      <CADEIA DE CARACTERES> /
      <VARIAVEL> /
      <REGISTRADOR>
<ENDERECO> ::= <ENDERECO DE PALAVRA> /
      <ENDERECO DE BYTE>
<ENDERECO DE PALAVRA> ::= @ <IDENTIFICADOR DE VETOR>
<ENDERECO DE BYTE> ::= % <IDENTIFICADOR DE VETOR>
<OPERADOR UNARIO> ::= + / - / NOT
<OPERADOR BINARIO> ::= <OPERADOR ARITMETICO> /
      <OPERADOR LOGICO> /
      <OPERADOR DE DESLOCAMENTO> /
      <OPERADOR DE ROTACAO>
<OPERADOR ARITMETICO> ::= + / - / * /
      <BARRA> / ! / MPY / DIV
<OPERADOR LOGICO> ::= AND / OR / XOR
<OPERADOR DE DESLOCAMENTO> ::= SHL / SHR / SHLA / SHRA
<OPERADOR DE ROTACAO> ::= RTL / RTR
```

A INSTRUCAO DE REGISTRADOR INDICA UMA SEQUENCIA DE OPERACOES, TRANSCORRIDAS NO REGISTRADOR INDICADO A ESQUERDA DO SIMBOLO := . AS OPERACOES SE PROCESSAM NA ORDEM EM QUE ESTAO ESCRITAS, DA ESQUERDA PARA A DIREITA; NAO HA PRECEDENCIA DE DETERMINADOS OPERADORES SOBRE OUTROS.

EM CADA CASO, DEVE-SE OBSERVAR O TIPO DO REGISTRADOR USADO E A NATUREZA DA OPERACAO, PARA DETERMINAR O TIPO CORRETO DO OPERANDO:

A) OPERADORES ARITMETICOS:
PARA OS OPERADORES DE SOMA (+), SUBTRACAO (-), MULTIPLICACAO (*), DIVISAO (/) E RESTO (!), O TIPO DO OPERANDO DEVE CORRESPONDER AO DO REGISTRADOR USADO. OPERACOES, NESSE CASO, TRANSCORRIDAS EM RA, NAO AFETAM O CONTEUDO DE RB, E VICE-VERSA.
O OPERADOR MPY (DE MULTIPLICACAO) SO SE APLICA A RA E DEVE SER SEGUIDO DE UM OPERANDO DE TIPO CURTO. RESULTA UM INTEIRO DUPLO EM RBA (ALTERANDO-SE POIS O CONTEUDO DE RB).
O OPERADOR DIV (DE DIVISAO) SO SE APLICA A RBA E DEVE SER SEGUIDO DE UM OPERANDO DE TIPO CURTO. RESULTA O QUOCIENTE EM RA E O RESTO EM RB.

B) OPERADORES LOGICOS:
APLICAM-SE A RA E RB (NAO A RBA) E SAO SEGUIDOS DE OPERANDOS DE TIPO CURTO. (AS OPERACOES LOGICAS COM RB GASTAM MAIS CODIGO E SAO MENOS RAPIDAS).

C) OPERADORES DE DESLOCAMENTO E ROTACAO:
APLICAM-SE A RA, RB E RBA, POSSIBILITANDO O "SHIFT" LOGICO (SHL,SHR), O "SHIFT" ARITMETICO (SHLA,SHRA) E A ROTACAO (RTL,RTR) DOS REGISTRADORES. PRECEDEM OPERANDOS DE TIPO CURTO.

OBSERVACOES ADICIONAIS:
A) AS CADEIAS USADAS COMO <TERMO> NAO DEVEM EXCEDER O COMPRIMENTO DE UMA PALAVRA.
B) SE D E' UMA VARIAVEL DUPLA, AS INSTRUCOES RA:=D E RB:=D SAO VALIDAS; INDICAM QUE SEJA CARREGADO NO REGISTRADOR REFERIDO O VALOR DOS 16 BITS DE MAIS BAIXA ORDEM DE D.
C) SE W E' UMA VARIAVEL CURTA, A INSTRUCAO RBA:=W E' VALIDA E INDICA QUE SEJA CARREGADO EM RBA O VALOR DE W, CONVERTIDO PARA 32 BITS.

EXEMPLOS:
SEJAM AS DECLARACOES:

```
WORD A,B,C,K,N
DOUBLE XX,YY,ZZ
WORD(20) VETOR
DOUBLE(15) CONTA
```

VALEM AS INSTRUCOES:

```
RA:= A+B *RB AND 3 +1
RB:= RB-2
RA:= NOT VETOR(C) RTL K OR 17B
RB:= YY-"0"
RA:= -N MPY 5
RBA:= +CONTA(RA) SHLA N DIV 48
RBA:= 0
RBA:= A DIV K
RBA:= XX+YY!ZZ
```

18. INSTRUCAO DE ARMAZENAMENTO

<INSTRUCAO DE ARMAZENAMENTO> ::=
<LISTA DE VARIAVEIS> := <REGISTRADOR>
<LISTA DE VARIAVEIS> ::= <VARIAVEL OU REGISTRADOR> /
<LISTA DE VARIAVEIS>, <VARIAVEL OU REGISTRADOR>
<VARIAVEL OU REGISTRADOR> ::= <VARIAVEL> / <REGISTRADOR>

A INSTRUCAO DE ARMAZENAMENTO DETERMINA QUE O VALOR DO REGISTRADOR A DIREITA DO SIMBOLO := SEJA COPIADO NAS VARIAVEIS QUE COMPOEM A LISTA. AS VARIAVEIS DEVEM CORRESPONDER EM TIPO AO REGISTRADOR INDICADO. A ATRIBUICAO DE VALOR SEGUE A ORDEM DA LISTA, DA ESQUERDA PARA A DIREITA.

EXEMPLOS:
K,N,VETOR(RA):=RB
RB,CEL:=RA
CONTADOR:=RBA

19. INSTRUCAO DE INCREMENTACAO

<INSTRUCAO DE INCREMENTACAO> ::= INCR <LISTA DE VARIAVEIS>

INCREMENTA DE UMA UNIDADE O VALOR DE CADA VARIAVEL DA LISTA. TODAS AS VARIAVEIS DEVEM SER DE TIPO CURTO E A INCREMENTACAO SEGUE A ORDEM DA LISTA, DA ESQUERDA PARA A DIREITA.

EXEMPLOS:
INCR RA
INCR RB,MEM(RB)

20. INSTRUCAO DE DESVIO

<INSTRUCAO DE DESVIO> ::= GOTO <IDENTIFICADOR DE ROTULO>

E' EXECUTADA A SEGUIR A INSTRUCAO QUE SE SEGUE A DEFINICAO DO ROTULO INDICADO.

EXEMPLO:
GOTO CRUZAMENTO

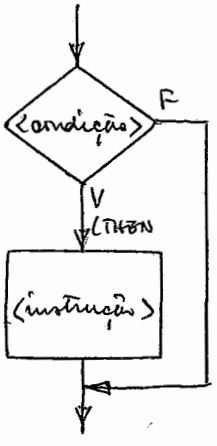
21. INSTRUCAO CONDICIONAL

<INSTRUCAO CONDICIONAL> ::= <PARTE IF> <INSTRUCAO> /
<PARTE IF> <INSTRUCAO> ELSE <INSTRUCAO>
<PARTE IF> ::= IF <CONDICAO> THEN
<CONDICAO> ::= <CONDICAO SIMPLES> /
<CONDICAO COMBINADA> /
<CONDICAO ALTERNATIVA>

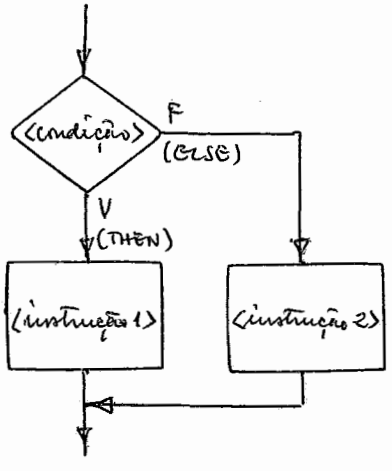
```
<CONDICAO COMBINADA> ::=
     <CONDICAO SIMPLES> AND <CONDICAO SIMPLES> /
     <CONDICAO COMBINADA> AND <CONDICAO SIMPLES>
<CONDICAO ALTERNATIVA> ::=
     <CONDICAO SIMPLES> OR <CONDICAO SIMPLES> /
     <CONDICAO ALTERNATIVA> OR <CONDICAO SIMPLES>
<CONDICAO SIMPLES> ::= <FLAG> / NOT <FLAG> /
     <BIT ESPECIAL> / NOT <BIT ESPECIAL> /
     <REGISTRADOR> <RELACAO> <TERMO DE RELACAO>
<RELACAO> ::= < / <= / = / >= / > / #
<TERMO DE RELACAO> ::= <CONSTANTE NUMERICA> /
     <CADEIA DE CARACTERES> /
     <VARIAVEL> /
     <REGISTRADOR>
```

O MECANISMO DA INSTRUCAO CONDICIONAL E' O SEGUINTE:

1) IF <CONDICAO> THEN <INSTRUCAO>



2) IF <CONDICAO> THEN <INSTRUCAO-1> ELSE <INSTRUCAO-2>



EXEMPLOS:

IF RB=VET(RA) THEN FINALIZACAO

```
    IF RB#TEMP THEN
       IF RA<"A" OR RA>"Z" THEN FURO
    IF OVFL THEN BEGIN MENS(1); RA:=-10 END
    IF NOT FLAG(0) THEN GOTO $ZERO
    IF RBA<0D THEN RBA:=0
    IF RA="/E" THEN IF RA=2 THEN FIM ELSE ERRO
```

OBSERVACOES:

1) O TERMO DA RELACAO DEVE CORRESPONDER EM TIPO AO REGISTRADOR TOMADO COMO REFERENCIA. AS CADEIAS DE CARACTERES NAO DEVEM EXCEDER O COMPRIMENTO DE UMA PALAVRA E SAO SUPOSTAS DE TIPO CURTO.
2) OS CASOS DE AMBIGUIDADE, COMO NO ULTIMO EXEMPLO ACIMA, RESOLVEM-SE SUPONDO A PARTE ELSE DE SIGNIFICADO INCERTO EM CORRESPONDENCIA COM A PARTE IF MAIS PROXIMA:

```
    IF RA="/E" THEN [IF RB=2 THEN FIM ELSE ERRO]
```

22. INSTRUCAO DE CHAVEAMENTO

```
    <INSTRUCAO DE CHAVEAMENTO> ::= <SEQUENCIA CASE>  <INSTRUCAO>
END
    <SEQUENCIA CASE> ::= <PARTE CASE> /
         <SEQUENCIA CASE> <INSTRUCAO>;
    <PARTE CASE> ::= CASE <INDICE> BEGIN
```

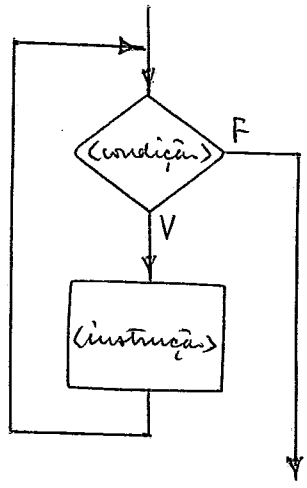
A INSTRUCAO DE CHAVEAMENTO PERMITE SELECIONAR UMA INSTRUCAO DENTRE UMA SEQUENCIA, SEGUNDO O VALOR DE UM INDICE. A SEQUENCIA E' SUPOSTA NUMERADA A PARTIR DE ZERO.

```
    EXEMPLO:
    CASE N
    BEGIN
       IF RB<REF THEN [RA:=X+5; X:=RA];
       IF RA<0 THEN LSUP:=RB ELSE LINF:=RB;
       ROTCASO2;
       ROTCASO3
    END
```

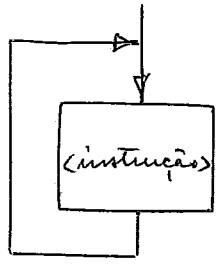
23. INSTRUCAO ITERATIVA

```
    <INSTRUCAO ITERATIVA> ::= <INSTRUCAO WHILE> /
         <INSTRUCAO REPEAT>
         <INSTRUCAO FOR>
    <INSTRUCAO WHILE> ::= WHILE <CONDICAO> DO <INSTRUCAO>
    <INSTRUCAO REPEAT> ::= REPEAT <INSTRUCAO> /
         REPEAT <INSTRUCAO> UNTIL <CONDICAO>
    <INSTRUCAO FOR> ::= FOR <VARIAVEL DE CONTROLE> DO
                  <INSTRUCAO>
    <VARIAVEL DE CONTROLE> ::= <VARIAVEL> / <REGISTRADOR>
```

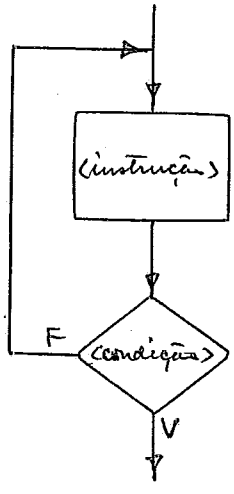
MECANISMO DA INSTRUCAO WHILE:
WHILE <CONDICAO> DO <INSTRUCAO>



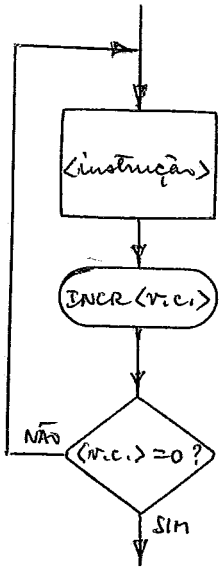
MECANISMO DA INSTRUCAO REPEAT:
FORMA 1: REPEAT <INSTRUCAO>



FORMA 2: REPEAT <INSTRUCAO> UNTIL <CONDICAO>



MECANISMO DA INSTRUCAO FOR:
FOR <VAR. CONTR.> DO <INSTRUCAO>



NOTA: A VARIAVEL DE CONTROLE DEVE SER DO TIPO CURTO.

EXEMPLOS DIVERSOS:
WHILE RB≤N DO BEGIN VETOR(RB):=RA; INCR RB END
REPEAT BEGIN VETOR(RB):=RA; INCR RB END UNTIL RB=N
REPEAT BEGIN ENTRADA; PROCESSAMENTO; SAIDA END
FOR K DO BEGIN RB:=K+N; VETOR(RB):=RA END

24. INSTRUCAO DE ENTRADA OU SAIDA

<INSTRUCAO DE E/S> ::= <OPERADOR DE E/S> <CANAL>
<OPERADOR DE E/S> ::= LIA / LIB /
MIA / MIB /
OTA / OTB /
HALT

AS OPERACOES DE ENTRADA OU SAIDA SAO DESIGNADAS PELOS MNEMONICOS DO "ASSEMBLER".

EXEMPLOS:
LIA 1B
OTA 5
HALT 31B

25. PROGRAMA

<PROGRAMA> ::= <PROGRAMA PRINCIPAL> /
<SEGMENTO> / <SUBPROGRAMA>

<PROGRAMA PRINCIPAL> ::= <BLOCO>
<SEGMENTO> ::= <BLOCO>
<SUBPROGRAMA> ::= <PROCEDIMENTO INTERNO>

TANTO O PROGRAMA PRINCIPAL COMO O SEGMENTO TEM A FORMA SINTATICA DE UM BLOCO; A DISTINCAO ENTRE AMBOS E' PROVIDA POR UM CARTAO DE CONTROLE (VER ADIANTE).

OS SUBPROGRAMAS TEM A FORMA SINTATICA DOS PROCEDIMENTOS INTERNOS, DEVENDO POREM SER DECLARADOS COMO PROCEDIMENTOS EXTERNOS NOS PROGRAMAS EM QUE SAO REFERENCIADOS.

26. PALAVRAS RESERVADAS

AS SEGUINTES PALAVRAS SAO CONSIDERADAS RESERVADAS, NAO PODENDO SER USADAS COMO IDENTIFICADORES:

AND	IF	RB
BEGIN	INCR	RB0
CASE	LABEL	RBA
CODE	LIA	REPEAT
COMMENT	LIB	RTL
CONTROL	MIA	RTR
DEF	MIB	SHL
DIV	MPY	SHLA
DO	NOT	SHR
DOUBLE	OR	SHRA
ELSE	OTA	THEN
END	OTB	UNTIL
EXT	OVFL	VALUE
FLAG	PROCEDURE	WHILE
FOR	RA	WORD
GOTO	RA0	XOR
HALT		

27. COMENTARIOS E BRANCOS

A FIM DE TORNAR MAIS LEGIVEIS OS PROGRAMAS, E' PERMITIDA A INCLUSAO DE COMENTARIOS, OBSERVADAS AS SEGUINTES REGRAS:

1) E' CONSIDERADO COMENTARIO QUALQUER TEXTO A PARTIR DO CARATER & (E COMERCIAL), ATE O FIM DO REGISTRO CORRENTE DO PROGRAMA-FONTE, DESDE QUE: A) ESSE CARATER NAO PERTENCA A UMA CADEIA; B) ESSE CARATER NAO FACA PARTE DE UM COMENTARIO ABERTO PELA PALAVRA RESERVADA COMMENT .

2) NO INICIO DO PROGRAMA OU APOS O SIMBOLO ; (PONTO E VIRGULA) OU APOS O SIMBOLO BEGIN , E' PERMITIDO INCLUIR COMENTARIOS DA FORMA:

COMMENT <QUALQUER TEXTO NAO CONTENDO ;> ;

3) IDENTIFICADORES, CONSTANTES NUMERICAS E CADEIAS DE CARACTERES PODEM SER USADOS COMO COMENTARIOS, IMEDIATAMENTE EM

SEGUIDA AO SIMBOLO END :

END <IDENTIFICADOR>
END <CONSTANTE NUMERICA>
END <CADEIA DE CARACTERES>
EQUIVALEM A END .

OS BRANCOS, DE MODO GERAL, PODEM SER INCLUIDOS A VONTADE NO TEXTO DO PROGRAMA. DEVE-SE NOTAR ENTRETANTO QUE:

1) UM BRANCO NO INTERIOR DE UMA CADEIA DE CARACTERES REPRESENTA O CARATER BRANCO.
2) DUAS PALAVRAS RESERVADAS ADJACENTES OU UMA PALAVRA RESERVADA E UM IDENTIFICADOR ADJACENTES DEVEM SER SEPARADOS POR PELO MENOS UM BRANCO.
3) NAO PODE HAVER BRANCOS ENTRE OS CARACTERES QUE COMPOEM UM IDENTIFICADOR, UMA PALAVRA RESERVADA OU UMA CONSTANTE NUMERICA.
4) NAO PODE HAVER BRANCO ENTRE OS CARACTERES QUE COMPOEM ALGUM DOS SIMBOLOS: := <= >= .

SO SAO CONSIDERADAS AS 72 PRIMEIRAS COLUNAS DE CADA REGISTRO-FONTE.

UM PROGRAMA E' SUPOSTO UMA SEQUENCIA CONTINUA DE CARACTERES, A SER ANALISADA DA ESQUERDA PARA A DIREITA. QUALQUER SIMBOLO DA LINGUAGEM COMPOSTO DE MAIS DE UM CARATER PODE SER QUEBRADO NA COLUNA 72 DE UM REGISTRO E CONTINUAR NA COLUNA 1 DO REGISTRO SEGUINTE.

28. IDENTIFICADORES PRE-DECLARADOS

A FIM DE FACILITAR A FEITURA DE PROGRAMAS EM LCS, E' FORNECIDA AO USUARIO UMA BIBLIOTECA DE ROTINAS UTEIS, TRATADAS COMO PRE-DECLARADAS PELO COMPILADOR. AS ROTINAS PRE-DECLARADAS DIFEREM DAS ROTINAS DITAS INTRINSECAS PELO FATO DE AS CHAMADAS AS INTRINSECAS SEREM INTRODUZIDAS PELO COMPILADOR A REVELIA DO AUTOR DO PROGRAMA, ENQUANTO QUE AS PRE-DECLARADAS SAO CHAMADAS DE FORMA EXPLICITA, DISPENSANDO-SE APENAS A SUA DECLARACAO COMO ROTINAS EXTERNAS.

AS ROTINAS PRE-DECLARADAS SAO DESCRITAS EM UM APENDICE A ESTE TRABALHO.

FINALMENTE, O COMPILADOR PRESSUPOE AINDA A DECLARACAO:

WORD(32768) MEM DEF 00000

E RESOLVE DE FORMA OTIMIZADA (SEM RECURSO A INDEXACAO) REFERENCIAS DO TIPO:

RA:=MEM(125B) OU

```
MEM(P):=RB                    OU AINDA
EXEC(2,1,MEM(P),-72) .
```

IV. PERSPECTIVAS DE EXTENSAO DA LINGUAGEM

O USO INTENSIVO DA LCS, NO SERPRO, NO DECORRER DE QUASE UM ANO E MEIO, REVELOU A CONVENIENCIA DE SE PROVER A LINGUAGEM DE TRES NOVOS RECURSOS: A DECLARACAO DE CONSTANTES, A DE VARIAVEIS COMUNS E O USO EXPLICITO DE VARIAVEIS DE 16 BITS COMO PONTEIROS. UM QUARTO RECURSO, QUE NUNCA CHEGOU A FAZER FALTA, MAS QUE PODERIA TORNAR-SE EVENTUALMENTE DE GRANDE UTILIDADE, SERIA A INTRODUCAO DO TIPO REAL , PARA TRATAMENTO DE OPERACOES DE PONTO FLUTUANTE.

APRESENTAMOS A SEGUIR UMA DISCUSSAO MAIS DETALHADA DAS EXTENSOES ACIMA PROPOSTAS.

1. DECLARACAO DE CONSTANTES

A DECLARACAO DE CONSTANTES PERMITE REPRESENTAR CONSTANTES POR IDENTIFICADORES. TRATA-SE DE EXCELENTE RECURSO PARA A PARAMETRIZACAO DE PROGRAMAS, NOS CASOS EM QUE DIVERSOS OPERANDOS, INDICES OU DIMENSOES PODEM MUDAR, DE COMPILACAO PARA COMPILACAO, NA DEPENDENCIA DE ALGUNS VALORES PRE-FIXADOS.

EM TERMOS FORMAIS, A EXTENSAO SE DARIA PELA SUBSTITUICAO DE TODA OCORRENCIA DA ENTIDADE SINTATICA <CONSTANTE NUMERICA> POR <VALOR NUMERICO> , E PELA INTRODUCAO DAS REGRAS:

```
<VALOR NUMERICO> ::= <CONSTANTE NUMERICA> /
     <IDENTIFICADOR DE CONSTANTE>
<IDENTIFICADOR DE CONSTANTE> ::= <IDENTIFICADOR>
<DECLARACAO> ::= <DECLARACAO DE CONSTANTE>
<DECLARACAO DE CONSTANTE> ::= CONSTANT <ITEM CONSTANTE> /
     <DECLARACAO DE CONSTANTE> , <ITEM CONSTANTE>
<ITEM CONSTANTE> ::= <IDENTIFICADOR> = <EXPRESSAO CONSTANTE>
<EXPRESSAO CONSTANTE> ::= <VALOR NUMERICO> /
     <EXPRESSAO CONSTANTE> <OPERADOR ARITMETICO>
          <VALOR NUMERICO>
```

O TIPO DA CONSTANTE ASSOCIADA A UM IDENTIFICADOR FICA PERFEITAMENTE DETERMINADO PELA <EXPRESSAO CONSTANTE> , NA QUAL OS VALORES CONSTANTES DEVEM SER USADOS COERENTEMENTE, COMO NAS EXPRESSOES DE REGISTRADOR.

EXEMPLOS:

```
CONSTANT DIM=100, ULT=DIM-1
CONSTANT ULTCAR=DIM*2-1
```

```
CONSTANT K=200000D, MK=K/2D

PODERIAMOS ENTAO ESCREVER:

WORD(DIM) AREA=DIM:"  ";
WORD ULTPAL DEF AREA(ULT);
...
RA:=-1; RB:=0;
REPEAT BEGIN
       RA:=RA XOR AREA(RB);
       INCR RB
       END
UNTIL RB=DIM;
```

2. DECLARACAO DE VARIAVEIS COMUNS

A DECLARACAO DE VARIAVEIS COMUNS E' BASTANTE UTIL PARA A DECOMPOSICAO DE UM PROGRAMA MUITO EXTENSO EM VARIOS SUBPROGRAMAS OU SEGMENTOS.

PODERIA SER ACRESCENTADA A LINGUAGEM PELA ADOCAO DAS SEGUINTES REGRAS:

<DECLARACAO DE VARIAVEIS> ::= <DECLARACAO DE AREA COMUM>
<DECLARACAO DE AREA COMUM> ::= COMMON <TIPO> <IDENTIFICADOR>
/
<DECLARACAO DE AREA COMUM>, <IDENTIFICADOR>

EXEMPLOS:

```
COMMON WORD X,Y
COMMON DOUBLE(60) LISTA
```

A FIM DE EVITAR A DECLARACAO DE MAIS DE UMA VARIAVEL EM COMMON, COM A MESMA IDENTIFICACAO, NO MESMO PROGRAMA, PODE-SE RESTRINGIR ESSE TIPO DE DECLARACAO AO BLOCO DE MAIS BAIXO NIVEL DE ENCAIXAMENTO.

3. PONTEIROS

A IMPLEMENTACAO DA IDEIA DE PONTEIROS TERIA POR FINALIDADE PERMITIR AO PROGRAMADOR A OTIMIZACAO DE REFERENCIAS A ELEMENTOS DE VETOR, NORMALMENTE REALIZADAS SOB A FORMA DE VARIAVEIS INDEXADAS.

NAO E' PARA ISSO NECESSARIO DEFINIR UM NOVO TIPO. BASTA PODERMOS DENOTAR UM OPERANDO ATRAVES DA VARIAVEL CUJO VALOR E' O

ENDERECO DESSE OPERANDO.

ADOTEMOS AS SEGUINTES REGRAS:

```
<PONTEIRO> ::= ( <VARIAVEL SIMPLES> ) /
     ( <REGISTRADOR> )
<TERMO> ::= <PONTEIRO>
<LISTA DE VARIAVEIS> ::= <PONTEIRO> /
     <LISTA DE VARIAVEIS>, <PONTEIRO>
<PARAMETRO EFETIVO> ::= <PONTEIRO>
```

O PONTEIRO DEVE SER REPRESENTADO POR UMA VARIAVEL SIMPLES OU REGISTRADOR, DE 16 BITS; EM SE TRATANDO DE VARIAVEL, SE FOR PARAMETRO FORMAL TERA DE TER SIDO ESPECIFICADO POR VALOR.

EXEMPLO:

SUPONHAMOS A ROTINA ZSNX, QUE ZERA OS ELEMENTOS NEGATIVOS DO VETOR X, DE 100 PALAVRAS:

```
PROCEDURE ZSNX;
   BEGIN
   RB:=0;
   REPEAT BEGIN
          RA:=X(RB);
          IF RA<0 THEN
             BEGIN RA:=0; X(RB):=RA END;
          INCR RB
          END
   UNTIL RB=100
   END ZSNX;
```

PODERIAMOS OTIMIZAR AS REFERENCIAS AO VETOR X PELO USO DE PONTEIROS:

```
PROCEDURE ZSNX;
   BEGIN
   WORD FIM;
   RB:=@X; RA:=RB+100; FIM:=RA;
   REPEAT BEGIN
          RA:=(RB);
          IF RA<0 THEN
             BEGIN RA:=0; (RB):=RA END;
          INCR RB
          END
   UNTIL RB=FIM
   END ZSNX;
```

ACRESCENTEMOS QUE A OTIMIZACAO ACIMA PODE SER HOJE OBTIDA PELO USO DO VETOR PRE-DECLARADO MEM; POREM, A INTRODUCAO DOS PONTEIROS ESTENDERIA O RECURSO AO TRATAMENTO DE CELULAS DE DUPLA PRECISAO, ALEM DE SER MAIS ELEGANTE, FORMALMENTE, QUE A ADOCAO

DE VETORES PRE-DECLARADOS DE TRATAMENTO ESPECIAL.

4. ARITMETICA DE PONTO FLUTUANTE

PARA A IMPLEMENTACAO DA ARITMETICA DE PONTO FLUTUANTE, SERIA NECESSARIO ACRESCENTAR A LINGUAGEM:

A) CONSTANTES DE PONTO FLUTUANTE, NA NOTACAO USUAL;
B) UM NOVO REGISTRADOR, RFL, DE PONTO FLUTUANTE, SUPERPOSTO A RBA;
C) UM NOVO TIPO, REAL, PARA A DECLARACAO DE VARIAVEIS E A ESPECIFICACAO DE PARAMETROS FORMAIS.

ASSIM, PODERIAMOS ESCREVER, POR EXEMPLO:

```
REAL RE1, RE2;
REAL(20) VETOR;
...
RFL:=VETOR(RA)-RE2*2.5-1.0E+3;
RE1:=RFL;
...
IF RFL>0.001 THEN ...
```

A CONVERSAO DE INTEIROS CURTOS E DUPLOS PARA REAIS, OU VICE-VERSA, PODERIA SER INDICADA DE MODO IMPLICITO, COMO ATUALMENTE A CONVERSAO DE INTEIROS CURTOS PARA DUPLOS OU DUPLOS PARA CURTOS.

SERIA AINDA PRECISO ACRESCENTAR A BIBLIOTECA DE PRE-DECLARADOS UMA ROTINA PARA A FORMATACAO DE REAIS EM CARACTERES ASCII E OUTRA PARA O INVERSO, CONFORME AS EXISTENTES PARA INTEIROS CURTOS E DUPLOS.

V. ESTRUTURA DO COMPILADOR IMPLEMENTADO

1. DADOS BASICOS

A VERSAO DISPONIVEL DO COMPILADOR LCS CORRESPONDE A DESCRICAO DA LINGUAGEM APRESENTADA NO CAPITULO III E COMPLEMENTADA PELOS APENDICES A ESTE TRABALHO.

O COMPILADOR FUNCIONA SOB CONTROLE DO DOS-HP (OU RTE-HP) E GERA CODIGO-OBJETO COMPATIVEL COM O "LOADER" FORNECIDO PELO FABRICANTE. TENDO SIDO ESCRITO BASICAMENTE EM ALGOL, ESSA PRIMEIRA VERSAO DO COMPILADOR OCUPA UM RAZOAVEL ESPACO DE MEMORIA (19K PALAVRAS) E, NA PRATICA, REQUER UMA CONFIGURACAO DE 32K PALAVRAS PARA O SEU FUNCIONAMENTO. A PROXIMA VERSAO, A SER DESENVOLVIDA EM 1977, SERA ESCRITA NA PROPRIA LINGUAGEM E DEVERA SER SUPORTADA POR UMA CONFIGURACAO MINIMA DE 16K - O QUE A TORNARA UTILIZAVEL PELA GRANDE MAIORIA DE SISTEMAS HP EM USO, 2100A OU 21MX. (OS SISTEMAS EM USO NO SERPRO TEM, QUASE TODOS, 24K DE MEMORIA; OS RESTANTES, 32K).

O COMPILADOR LCS E' UM COMPILADOR DE UM SO PASSO; ISTO E', EM UM SO PERCORRIMENTO DO PROGRAMA ELE LISTA OS REGISTROS-FONTE, RECONHECE OS SIMBOLOS UTILIZADOS, REALIZA A ANALISE SINTATICA, CONSTROI A TABELA DE SIMBOLOS, RESOLVE AS REFERENCIAS, ASSINALA OS ERROS EVENTUAIS E GERA EM DISCO O CODIGO-OBJETO.

A IDEIA DE CONSTRUIR UM COMPILADOR DE UM SO PASSO ANTECEDEU A PROPRIA DEFINICAO PRECISA DA LINGUAGEM, E SE ORIGINOU DO FATO DE NAO SER NECESSARIA UMA ETAPA A PARTE DE OTIMIZACAO DE CODIGO. DE FATO, POR SER A LCS UMA LINGUAGEM DE MEDIO NIVEL, O COMPILADOR ENCARREGA-SE APENAS DA OTIMIZACAO MAIS IMEDIATA - POR EXEMPLO, A DE ESCOLHER UM CODIGO MAIS RAPIDO OU MAIS COMPACTO PARA OS CASOS PARTICULARES DE UMA DADA OPERACAO -, FICANDO RELEGADO AO PROPRIO PROGRAMADOR A TAREFA DE OTIMIZAR GLOBALMENTE O PROGRAMA.

A NIVEL DA DEFINICAO DA LINGUAGEM, A COMPILACAO EM UM SO PASSO E' FACILITADA PELA INTRODUCAO DA DECLARACAO DE ROTULO E PELA OBRIGATORIEDADE DA DECLARACAO DE QUALQUER IDENTIFICADOR ANTES DE QUALQUER REFERENCIA A ELE. (EXCETUAM-SE A ESTA ULTIMA REGRA APENAS OS IDENTIFICADORES SUPOSTOS PRE-DECLARADOS PELO COMPILADOR).

2. ESTRUTURA BASICA

A ESTRUTURA DO COMPILADOR PODERIA, A GROSSO MODO, SER DESCRITA:

```
BEGIN
STRING(20) SIMB;
INTEGER COD;
REPEAT BEGIN
       ANALISELEXICA(SIMB,COD);
       ANALISESINTATICA(SIMB,COD)
       END
UNTIL SIMB=""
END LCS
```

(ALGORITMO EM DIALETO ALGOLW).

3. ANALISADOR LEXICO

O PAPEL DO ANALISADOR LEXICO E' EXAMINAR O TEXTO-FONTE E DEVOLVER OS SIMBOLOS TERMINAIS DO PROGRAMA. CABE A ESSA ROTINA O RECONHECIMENTO DE COMENTARIOS (CUJA EXISTENCIA DEVE SER IGNORADA PELO ANALISADOR SINTATICO) E A OBTENCAO DE ALGUMAS ENTIDADES SINTATICAS A SEREM TRATADAS COMO SIMBOLOS TERMINAIS PELO ANALISADOR SINTATICO: IDENTIFICADORES, CONSTANTES NUMERICAS E CADEIAS DE CARACTERES. E' AINDA DE SUA COMPETENCIA COMANDAR A LISTAGEM DO PROGRAMA-FONTE E ASSINALAR ALGUNS ERROS - ESPECIFICAMENTE, AQUELES QUE DIZEM RESPEITO APENAS AO RECONHECIMENTO DOS SIMBOLOS, ABSTRAIDO O CONTEXTO, ISTO E', A SEQUENCIA EM QUE ESSES APARECEM NO PROGRAMA.

NO ESQUEMA ACIMA APRESENTADO, O ANALISADOR LEXICO RETORNA EM SIMB A REPRESENTACAO EM CARACTERES DO SIMBOLO DETECTADO E EM COD UM CODIGO NUMERICO RELATIVO A ESSE SIMBOLO, PARA USO DO ANALISADOR SINTATICO. EM CASO DE FIM DE PROGRAMA, TERIAMOS, POR EXEMPLO, SIMB="" (CADEIA VAZIA) E COD=0.

A ESTRUTURA DO ANALISADOR LEXICO PODE SER ASSIM DELINEADA:

```
PROCEDURE ANALISELEXICA(STRING(20) RESULT SIMB;
                        INTEGER RESULT COD);
  BEGIN
  OWN BOOLEAN TRAVA;
  OWN STRING(1) CARATER;
  OWN INTEGER CLASSE;
  IF TRAVA THEN TRAVA:= FALSE
  ELSE PROXIMOCARATER(CARATER,CLASSE);
  WHILE CLASSE<0
  DO PROXIMOCARATER(CARATER,CLASSE);
  CASE CLASSE+1 OF
     BEGIN
     FIMDOPROGRAMA;
     LETRA;
     ALGARISMO;
     ASPA;
```

```
    ...
    END
END ANALISELEXICA;
```

O MECANISMO E' O SEGUINTE:
O ANALISADOR CONSULTA A VARIAVEL LOGICA TRAVA, A QUAL E' SUPOSTA INICIALIZADA COM O VALOR FALSE E INDICA SE O ULTIMO CARATER DETECTADO NA ULTIMA CHAMADA DEVE AINDA SER CONSIDERADO NA ATIVACAO CORRENTE. CASO AFIRMATIVO, A VARIAVEL TRAVA E' SIMPLESMENTE DESLIGADA; CASO CONTRARIO E' PEDIDO O PROXIMO CARATER DO PROGRAMA-FONTE. EM SEGUIDA, O ANALISADOR AVANCA ATE O PROXIMO CARATER VALIDO E NAO BRANCO. ESTE SERA O PRIMEIRO CARATER DO SIMBOLO DETECTADO; CONFORME A SUA "CLASSE", SERA ATIVADA UMA ROTINA DO ANALISADOR, QUE COMPLETARA O RECONHECIMENTO DO SIMBOLO. CASO SEJA DETECTADO UM COMENTARIO, O PROCESSO E' REINICIADO.

A ROTINA PROXIMOCARATER CABE A OBTENCAO DO PROXIMO CARATER DO PROGRAMA-FONTE E A CLASSIFICACAO DESSE CARATER. CLASSE<0 INDICA BRANCO; CLASSE=0, FIM DE PROGRAMA; CLASSE=1, LETRA, E ASSIM POR DIANTE. CARACTERES INVALIDOS SAO TRATADOS COMO BRANCOS, SENDO POREM O PROGRAMADOR ADVERTIDO DE SUA PRESENCA.

4. ANALISADOR SINTATICO

O ANALISADOR SINTATICO TEM POR FUNCAO EXAMINAR A VALIDADE DA SEQUENCIA DE SIMBOLOS QUE COMPOE O PROGRAMA, COMANDANDO, AO MESMO TEMPO, AS ROTINAS QUE CONSTROEM E CONSULTAM A TABELA DE SIMBOLOS E AS QUE GERAM O CODIGO-OBJETO.

O METODO ADOTADO PARA SUA CONSTRUCAO E' O PROPOSTO POR DAVID GRIES PARA AS GRAMATICAS DE OPERADORES, UTILIZANDO MATRIZES DE TRANSICAO (*4 E *5). NOTANDO-SE QUE OS IDENTIFICADORES, CONSTANTES E CADEIAS DE CARACTERES SAO TRATADOS COMO SIMBOLOS TERMINAIS PELO ANALISADOR SINTATICO, E' FACIL ESCREVER UMA GRAMATICA DE OPERADORES EQUIVALENTE A ANTERIORMENTE APRESENTADA PARA A LCS. NA VERDADE, A PROPRIA DEFINICAO DA LINGUAGEM REFLETE O PROPOSITO DE FACILITAR ESSA TRANSFORMACAO.

AS GRAMATICAS LIVRES DE CONTEXTO DITAS DE OPERADORES SAO, POR DEFINICAO, AS EM QUE NENHUMA FORMA SENTENCIAL CONTEM DOIS SIMBOLOS NAO-TERMINAIS CONSECUTIVOS. DEMONSTRA-SE QUE UMA GRAMATICA LIVRE DE CONTEXTO E' UMA GRAMATICA DE OPERADORES SE E SOMENTE SE NO LADO DIREITO DE NENHUMA PRODUCAO APARECEM DOIS SIMBOLOS NAO-TERMINAIS CONSECUTIVOS.

O METODO PROPOSTO POR D. GRIES CONSISTE EM OBTER UMA GRAMATICA EQUIVALENTE A GRAMATICA DE OPERADORES ORIGINAL PELA INTRODUCAO DE NOVOS SIMBOLOS NAO-TERMINAIS, DITOS NAO-TERMINAIS ESTRELADOS, DE MODO A QUE TODA PRODUCAO DA GRAMATICA

TRANSFORMADA SE ENQUADRE EM AGUMA DAS FORMAS ABAIXO:

U' ::= T	U::=U'
U' ::= U'T	U ::= U'U
U' ::= UT	U ::= U
U' ::= U'UT	

ONDE T REPRESENTA UM SIMBOLO TERMINAL, U' UM NAO-TERMINAL ESTRELADO E U UM NAO-TERMINAL DA GRAMATICA PRIMITIVA. A GRAMATICA TRANSFORMADA E' DITA DE OPERADORES AUMENTADA.

A ORGANIZACAO DO ANALISADOR SINTATICO DERIVA DIRETAMENTE DA GRAMATICA DE OPERADORES AUMENTADA. O ALGORITMO UTILIZA UMA PILHA PARA OS NAO-TERMINAIS ESTRELADOS E UMA VARIAVEL PARA A REPRESENTACAO DO NAO-TERMINAL (NAO-ESTRELADO) CORRENTE. AS LINHAS DA MATRIZ DE TRANSICAO CORRESPONDEM AOS NAO-TERMINAIS ESTRELADOS E AS COLUNAS AO SIMBOLO TERMINAL CORRENTE. UM SIMBOLO NAO-TERMINAL ESTRELADO E' REPRESENTADO NA PILHA PELO NUMERO DE LINHA CORRESPONDENTE NA MATRIZ.

A CADA PASSO DA ANALISE, O ESTRELADO DE TOPO NA PILHA E O SIMBOLO TERMINAL CORRENTE FORNECEM UM ELEMENTO DA MATRIZ, A CUJO VALOR CORRESPONDE UNIVOCAMENTE UMA ROTINA A SER ATIVADA. A ROTINA TOMA AS PROVIDENCIAS ADEQUADAS, OBSERVADO O NAO-TERMINAL NAO-ESTRELADO CORRENTE, EFETUANDO UMA DAS SETE REDUCOES POSSIVEIS (NA VERDADE, SEIS, JA QUE A GRAMATICA DE OPERADORES PODE SER ALIVIADA DE PRODUCOES DA FORMA U::= U). NOTEMOS QUE A REDUCAO A SE PROCESSAR DEPENDE APENAS DO ELEMENTO DE TOPO DA PILHA DE NAO-TERMINAIS ESTRELADOS, DO NAO-TERMINAL NAO-ESTRELADO CORRENTE E DO SIMBOLO TERMINAL CORRENTE.

PODERIAMOS CITAR COMO AS QUALIDADES MAIS NOTAVEIS DO METODO ACIMA: A VELOCIDADE DO ANALISADOR, A FACILIDADE DE PROGRAMA-LO E A SIMPLICIDADE DE IMPLEMENTAR A RECUPERACAO DE ERROS. O USO DA MATRIZ DE TRANSICAO, INDUZINDO A DECOMPOSICAO DO ANALISADOR EM DIVERSAS ROTINAS, CADA UMA ENCARREGADA DO TRATAMENTO DE UMA SITUACAO PRECISA E BEM DETERMINADA, FACILITA GRANDEMENTE A CONSTRUCAO DO COMPILADOR; O PROGRAMADOR PODE-SE CONCENTRAR DE CADA VEZ NA IMPLEMENTACAO DE UM RECURSO E DESENVOLVER GRADUALMENTE O TRABALHO, CONSTRUINDO O COMPILADOR PARA SUBCONJUNTOS CRESCENTES DA LINGUAGEM.

A DESVANTAGEM DO METODO, SEGUNDO O PROPRIO AUTOR, SERIA O ESPACO OCUPADO PELO ANALISADOR - UM TANTO EXTENSO SE COMPARADO COM O DE OUTROS ESQUEMAS.

MAIS ESPECIFICAMENTE, EM TERMOS ALGORITMICOS, PODERIAMOS ASSIM APRESENTAR A ESTRUTURA DO ANALISADOR, CONFORME PROPOSTA POR D. GRIES:

```
PROCEDURE ANALISESINTATICA(STRING(20) VALUE SIMB;
                           INTEGER VALUE T);
  BEGIN
  OWN INTEGER U;
  OWN INTEGER P;
```

```
OWN INTEGER ARRAY PILHA[0:63];
CASE MATRIZ[PILHA[P],T]-1 OF
   BEGIN
   ERRO;
   ROTINA.001;
   ROTINA.002;
   ...
   END
END ANALISESINTATICA;
```

NA PROGRAMACAO DO COMPILADOR LCS, PREFERIMOS, POREM, ALTERAR LIGEIRAMENTE ESSE ESQUEMA PARA O SEGUINTE:

```
PROCEDURE ANALISESINTATICA(STRING(20) VALUE SIMB;
                           INTEGER VALUE T);
  BEGIN
  OWN INTEGER U;
  OWN INTEGER P;
  OWN INTEGER ARRAY PILHA[0:63];
  CASE PILHA[P] OF
     BEGIN
     ESTRELADO.01;
     ESTRELADO.02;
     ...
     END
  END ANALISESINTATICA;
```

ISTO E', ACRESCENTAMOS A ESTRUTURA DO PROGRAMA UM CONJUNTO DE ROTINAS CORRESPONDENTE AO CONJUNTO DE NAO-TERMINAIS ESTRELADOS DA GRAMATICA DE OPERADORES AUMENTADA. EM CADA UMA DESSAS ESTARA DETERMINADO O QUE FAZER FACE AO SIMBOLO TERMINAL RECEBIDO.

ENCONTRAMOS NESSE ESQUEMA ALTERNATIVO DUAS VANTAGENS IMPORTANTES:

A) E' FREQUENTE OCORRER QUE, ACHANDO-SE NO TOPO DA PILHA UM CERTO NAO-TERMINAL ESTRELADO, SEJA ACEITAVEL APENAS UM OU ALGUNS POUCOS SIMBOLOS TERMINAIS. A ROTINA CORRESPONDENTE A ESSE ESTRELADO PODERA ENTAO SER ESCRITA SIMPLESMENTE SOB A FORMA DE UMA INSTRUCAO CONDICIONAL:

```
IF T= SIMBOLO ESPERADO
THEN ACAO CABIVEL NESTE CASO
ELSE ERRO
```

(SENDO ERRO UMA ROTINA GERAL PARA TRATAMENTO DE ERROS SINTATICOS).

EM CONSEQUENCIA, ECONOMIZA-SE UMA NOVA INSTRUCAO CASE, CORRESPONDENTE A UMA LINHA DA MATRIZ DE TRANSICAO, E O ANALISADOR FICA MAIS LEGIVEL E COMPACTO.

B) A RAMIFICACAO POR NAO-TERMINAL ESTRELADO FACILITA ACOMPANHAR A LOGICA DA ANALISE SINTATICA E PORTANTO A DEPURACAO

DO COMPILADOR E A INCORPORACAO DE NOVOS RECURSOS A LINGUAGEM. A ADOCAO DA MATRIZ DE TRANSICAO PROPRIAMENTE DITA INDUZ A UMA CERTA DILUICAO DA ESTRUTURA DO PROGRAMA, A UMA CERTA DISPERSIVIDADE.

A GRAMATICA DE OPERADORES AUMENTADA, EM QUE SE BASEOU O ANALISADOR PARA A LCS, COMPOEM-SE DE 78 NAO-TERMINAIS ESTRELADOS, 23 NAO-TERMINAIS NAO-ESTRELADOS E 27 SIMBOLOS TERMINAIS. A PILHA E' SUPOSTA INICIALIZADA POR UM ESTRELADO ARTIFICIALMENTE INTRODUZIDO NA LINGUAGEM, <LAMBDA*>. A CELULA DE NAO-TERMINAIS ESTA' DE INICIO VAZIA (VALOR ZERO). E A COMPILACAO E' ENCERRADA NORMALMENTE AO SE CHEGAR A UMA DAS SITUACOES:

<LAMBDA*> <BLOCO> FIM
<LAMBDA*> <PROCEDIMENTO INTERNO> FIM

(FIM E' O SIMBOLO TERMINAL QUE INDICA FINAL DO PROGRAMA-FONTE).

APESAR DE TER SIDO ESCRITO EM ALGOL, O COMPILADOR PODE SER CONSIDERADO BASTANTE RAPIDO, COMPARADO AOS DEMAIS FORNECIDOS PELO FABRICANTE: PARA PROGRAMAS TIPICOS DE MAIS DE 3000 CARTOES, COMPILA A VELOCIDADE DE 740 CARTOES POR MINUTO (PROGRAMA-FONTE EM DISCO; LISTAGEM SUPRIMIDA E GERACAO DE CODIGO NORMAL).

5. RECUPERACAO DE ERROS

DO PONTO DE VISTA DA RECUPERACAO, OS ERROS PODEM SER GRUPADOS EM TRES CATEGORIAS:

A) ERROS QUE IMPEDEM O PROSSEGUIMENTO DA COMPILACAO. EXEMPLOS: AREA DE TRABALHO NA MEMORIA EXCEDIDA, CADEIA DE CARACTERES INTERROMPIDA POR FIM DE PROGRAMA. O COMPILADOR SIMPLESMENTE ASSINALA O EVENTO E DEVOLVE O CONTROLE AO SISTEMA.

B) ERROS DE RECUPERACAO IMEDIATA: FREQUENTEMENTE, ERROS SEMANTICOS. EXEMPLOS: PARAMETRO EFETIVO INCOMPATIVEL COM O PARAMETRO FORMAL DECLARADO; NUMERO OCTAL INVALIDO; DIMENSAO INVALIDA. O COMPILADOR INDICA O ERRO E VAI EM FRENTE, SEM PREJUIZO DA ANALISE SINTATICA, SENDO APENAS NECESSARIO, EM ALGUNS CASOS, FAZER ALGUM AJUSTE SEMANTICO; POR EXEMPLO, SE A DIMENSAO ATRIBUIDA A UM VETOR E' INVALIDA, SUPO-LA IGUAL A 1.

C) ERROS QUE CONTRARIAM A SINTAXE DA LINGUAGEM. PARA PODER PROSSEGUIR A COMPILACAO, O ANALISADOR DEVE LIVRAR-SE DO TRECHO EM QUE OCORREU O ERRO, PROCURANDO EVITAR O ASSINALAMENTO DE NOVOS ERROS QUE SEJAM MERO REFLEXO DO ANTERIOR. O COMPILADOR DISPOE PARA TANTO DE UMA ROTINA GERAL DE RECUPERACAO DE ERRO, A QUAL ESVAZIA A CELULA DE NAO-TERMINAIS NAO-ESTRELADOS, AVANCA NO PROGRAMA-FONTE ATE OBTER UM SIMBOLO TERMINAL CONSIDERADO

IMPORTANTE E DESBASTA A PILHA DE NAO-TERMINAIS ESTRELADOS ATE ENCONTRAR UM SIMBOLO COMPATIVEL COM O TERMINAL IMPORTANTE DETECTADO. OS TERMINAIS QUE SERVEM DE DELIMITADORES PARA A RECUPERACAO DE ERRO SAO: BEGIN END ; .

6. CARTOES DE CONTROLE DO COMPILADOR

OS CARTOES DE CONTROLE CARACTERIZAM-SE POR UMA INTERROGACAO (?) NA PRIMEIRA COLUNA.

A) CARTAO LCS

DEVE SER O PRIMEIRO CARTAO DO PROGRAMA-FONTE E SO PODE OCORRER UMA VEZ:

```
?LCS,<NOME DO PROGRAMA>
?LCS,<TIPO DO PROGRAMA>,<NOME DO PROGRAMA>
```

<NOME DO PROGRAMA> ::= <CADEIA DE CARACTERES>
<TIPO DO PROGRAMA> ::= M / P / S

O NOME DO PROGRAMA E' O QUE O IDENTIFICARA JUNTO AO SISTEMA. O TIPO PERMITE A SUA CLASSIFICACAO COMO PROGRAMA-MESTRE OU PRINCIPAL (M), PROCEDIMENTO (P), OU SEGMENTO (S). NAO SENDO INDICADO UM TIPO, TRATA-SE DE UM PROGRAMA PRINCIPAL.

B) CARTAO PARA SALTO DE PAGINA

```
?P
```

DETERMINA UM SALTO DE PAGINA NA UNIDADE DE LISTAGEM (SE O PROGRAMA ESTIVER SENDO LISTADO).

C) CARTOES PARA LISTAGEM DO PROGRAMA-FONTE

```
?L
?NL
```

O PRIMEIRO (?L) DETERMINA QUE O PROGRAMA-FONTE SEJA LISTADO DURANTE A COMPILACAO. O SEGUNDO (?NL) DESFAZ O EFEITO DO PRIMEIRO, E VICE-VERSA, DE MODO QUE E' POSSIVEL LISTAR APENAS TRECHOS SELECIONADOS DO PROGRAMA-FONTE. NADA SENDO INDICADO, O PROGRAMA E' LISTADO.

D) CARTOES PARA LISTAGEM DO CODIGO OBJETO GERADO

```
?C
?NC
```

EM PRINCIPIO, APENAS PARA DEPURACAO DO COMPILADOR. SEMELHANTES A ?L E ?NL; APENAS, SE NADA INDICADO, O CODIGO-OBJETO NAO E' LISTADO.

E) CARTOES PARA RASTREAMENTO DA ANALISE SINTATICA

```
?S
?NS
```

SEMELHANTES AOS ANTERIORES.

APENDICE I

BIBLIOTECA DE ROTINAS PRE-DECLARADAS

1) PROCEDURE LDBYTE(WORD(*) VETOR, WORD VALUE N);
   CODE

E' COPIADO NO BYTE DIREITO DE RA O VALOR DO BYTE DE ORDEM N DE VETOR; O BYTE ESQUERDO DE RA E' ZERADO.

2) PROCEDURE STBYTE(WORD(*) VETOR, WORD VALUE N);
   CODE

E' COPIADO NO BYTE DE ORDEM N DE VETOR O CONTEUDO DO BYTE DIREITO DE RA; O VALOR DE RA NAO SE ALTERA.

3) PROCEDURE MVBYTE(WORD VALUE N,
   WORD(*) ORIGEM, WORD VALUE PO,
   WORD(*) DESTINO; WORD VALUE PD);
   CODE

COPIA SOBRE O VETOR DESTINO, A PARTIR DO BYTE DE ORDEM PD, A CADEIA DE N BYTES SITUADA SOBRE O VETOR ORIGEM, A PARTIR DO BYTE DE ORDEM PO.

4) PROCEDURE CPBYTE(WORD VALUE N,
   WORD(*) CADEIA1, WORD VALUE P1,
   WORD(*) CADEIA2, WORD VALUE P2);
   CODE

COMPARA DUAS CADEIAS DE COMPRIMENTO N; UMA SOBRE CADEIA1, A PARTIR DO BYTE DE ORDEM P1, OUTRA SOBRE CADEIA2, A PARTIR DO BYTE DE ORDEM P2. O RESULTADO RETORNA EM RA:

A) SE RA<0, A PRIMEIRA CADEIA E' MENOR QUE A SEGUNDA;
B) SE RA=0, AS CADEIA SAO IGUAIS;
C) SE RA>0, A PRIMEIRA CADEIA E' MAIOR QUE A SEGUNDA.

A COMPARACAO E' FEITA BYTE A BYTE, DA ESQUERDA PARA A DIREITA, CADA BYTE SUPOSTO POSITIVO.

5) PROCEDURE MOVE(WORD VALUE N,
   WORD(*) ORIGEM, WORD VALUE PO,
   WORD(*) DESTINO, WORD VALUE PD);
   CODE

COPIA, SOBRE O VETOR DESTINO, A PARTIR DA PALAVRA DE ORDEM PD, A SEQUENCIA DE N PALAVRAS SITUADA SOBRE O VETOR ORIGEM, A PARTIR DA PALAVRA DE ORDEM PO.

```
6)   PROCEDURE COMPARE(WORD VALUE N,
        WORD(*) VETOR1, WORD VALUE P1,
        WORD(*) VETOR2, WORD VALUE P2);
        CODE
```

COMPARA DUAS SEQUENCIAS DE N PALAVRAS; A PRIMEIRA SITUADA SOBRE VETOR1, A PARTIR DA PALAVRA DE ORDEM P1, A SEGUNDA SITUADA SOBRE VETOR2, A PARTIR DA PALAVRA DE ORDEM P2.
RESULTADO EM RA, DA MESMA FORMA QUE PARA A ROTINA CPBYTE.

```
7)   PROCEDURE BIT(WORD(*) BITS, WORD VALUE N);
        CODE
```

COLOCA NO BIT DE EXTENSAO (EXT) O VALOR DO BIT DE ORDEM N DO VETOR BITS.

```
8)   PROCEDURE SET(WORD(*) BITS, WORD VALUE N);
        CODE
```

ACENDE O BIT DE ORDEM N DO VETOR BITS.

```
9)   PROCEDURE CLEAR(WORD(*) BITS, WORD VALUE N);
        CODE
```

APAGA O BIT DE ORDEM N DO VETOR BITS.

```
10)   PROCEDURE LDEO(WORD VALUE EO);
         CODE
```

ATRIBUI AOS BITS EXT E OVFL DA MAQUINA, RESPECTIVAMENTE, O ESTADO DOS BITS 15 E 0 DE EO.

```
11)   PROCEDURE STEO(WORD EO);
         CODE
```

ATRIBUI AOS BITS 15 E 0 DE EO, RESPECTIVAMENTE, O ESTADO DOS BITS EXT E OVFL DA MAQUINA. OS DEMAIS BITS DE EO SAO ZERADOS.

```
12)  PROCEDURE SETLINK(PROCEDURE ROTINA,
        WORD VALUE CANAL,LIGACAO);
        CODE
```

ATRIBUI A ROTINA A TAREFA DE PROCESSAR AS INTERRUPCOES CORRESPONDENTES AO CANAL ESPECIFICADO. A VARIAVEL LIGACAO FORNECE UMA POSICAO DA PAGINA DE BASE ONDE SERA ARMAZENADO O ENDERECO DA ROTINA.

13) PROCEDURE DECBIN(WORD(*) AREA, WORD VALUE P,N);
CODE

DEVOLVE EM RA O VALOR DO INTEIRO REPRESENTADO EM CODIGO ASCII NO VETOR AREA, POR UMA CADEIA DE N BYTES, A PARTIR DO BYTE DE ORDEM P. NO RETORNO, BIT DE EXTENSAO LIGADO INDICA ERRO DE REPRESENTACAO E O DE "OVERFLOW", IMPOSSIBILIDADE DE CONVERSAO PARA 16 BITS. A REPRESENTACAO DO NUMERO PODE SER PRECEDIDA DE UM SINAL E ANTECEDIDA OU SUCEDIDA DE ESPACOS.

14) PROCEDURE OCTBIN(WORD(*) AREA, WORD VALUE P,N);
CODE

SEMELHANTE A DECBIN, PARA A REPRESENTACAO OCTAL EM ASCII DE UM NUMERO.

15) PROCEDURE ASCBIND(WORD(*) AREA, WORD VALUE P,N);
CODE

SEMELHANTE A DECBIN: DEVOLVE EM RBA O VALOR DO INTEIRO DUPLO REPRESENTADO EM ASCII.

16) PROCEDURE BINDEC(WORD(*) AREA, WORD VALUE P,N);
CODE

COLOCA, NO CAMPO DE N BYTES DO VETOR AREA, A PARTIR DO BYTE DE ORDEM P, A REPRESENTACAO DECIMAL EM ASCII DO INTEIRO PASSADO EM RA. SAO SUPRIMIDOS OS ZEROS NAO SIGNIFICATIVOS E O NUMERO E' POSICIONADO A DIREITA DO CAMPO. ESTE SERA PREENCHIDO COM ASTERISCOS, CASO NAO POSSA ACOMODAR A CADEIA DESEJADA.

17) PROCEDURE BINOCT(WORD(*) AREA, WORD VALUE P,N);
CODE

SEMELHANTE A BINDEC, FORNECENDO A REPRESENTACAO OCTAL EM ASCII DO VALOR DE RA.

18) PROCEDURE BINDASC(WORD(*) AREA, WORD VALUE P,N);
CODE

SEMELHANTE A BINDEC, FORNECENDO A REPRESENTACAO EM ASCII DO

INTEIRO DUPLO DADO EM RBA.

19) PROCEDURE DEC(WORD(*) AREA, WORD VALUE P,N);
CODE

SEMELHANTE A BINDEC, FORNECENDO A REPRESENTACAO EM ASCII, POREM SEM SUPRESSAO DE ZEROS, DO INTEIRO DADO EM RA (BIT DE SINAL TOMADO TAMBEM COMO DE MAGNITUDE).

20) PROCEDURE OCT(WORD(*) AREA, WORD VALUE P,N);
CODE

SEMELHANTE A BINOCT, FORNECENDO A REPRESENTACAO EM ASCII, SEM SUPRESSAO DE ZEROS, DO INTEIRO DADO EM RA.

21) PROCEDURE ASC(WORD(*) AREA, WORD VALUE P,N);
CODE

SEMELHANTE A BINDASC, FORNECENDO A REPRESENTACAO EM ASCII, SEM SUPRESSAO DE ZEROS, DO INTEIRO DUPLO DADO EM RBA.

22) PROCEDURE FILL(WORD(*) AREA, WORD VALUE P,N);
CODE

PREENCHE O CAMPO DE N BYTES, LOCALIZADO SOBRE O VETOR AREA, A PARTIR DO BYTE DE ORDEM P, COM O VALOR DO BYTE DIREITO DE RA. SE N E' NULO, NENHUM BYTE E' COPIADO.

23) PROCEDURE SCAN(WORD(*) AREA, WORD VALUE P,N);
CODE

PROCURA, DA ESQUERDA PARA A DIREITA, NO CAMPO DE N BYTES DO VETOR AREA, A PARTIR DO BYTE DE ORDEM P, UM BYTE IGUAL OU DIFERENTE DO BYTE DIREITO DE RA: SE O BIT 15 DE RA ESTIVER ACESO, E' PESQUISADO UM BYTE DIFERENTE DO DIREITO DE RA; SE APAGADO, IGUAL. RETORNA EM RB O NUMERO DE ORDEM (NO VETOR) DO PRIMEIRO BYTE ENCONTRADO, OU -1 CASO A PESQUISA SEJA INFRUTIFERA.

24) PROCEDURE SCANR(WORD(*) AREA, WORD VALUE P,N);
CODE

SEMELHANTE A SCAN, SENDO POREM A PESQUISA REALIZADA DA DIREITA PARA A ESQUERDA, NO CAMPO ESPECIFICADO.

25) PROCEDURE DESCR(WORD(*) AREA);
CODE

DEVOLVE EM RA O ENDERECO DA PRIMEIRA POSICAO DO VETOR AREA E EM RB, COM O SINAL TROCADO, O NUMERO DE PALAVRAS DO VETOR.

26) PROCEDURE LBT;
CODE

CARREGA NO BYTE DIREITO DE RA O VALOR DO BYTE CUJO ENDERECO E' DADO EM RB. O BYTE ESQUERDO DE RA E' ZERADO E O VALOR DE RB ACRESCIDO DE UMA UNIDADE.

27) PROCEDURE SBT;
CODE

ARMAZENA NO BYTE DE ENDERECO DADO EM RB O VALOR DO BYTE DIREITO DE RA. O VALOR DE RB E' ACRESCIDO DE UMA UNIDADE.

28) PROCEDURE MBT(WORD VALUE N);
CODE

MOVE A CADEIA DE N BYTES, LOCALIZADA A PARTIR DO BYTE DE ENDERECO FORNECIDO EM RA, PARA A DE ENDERECO FORNECIDO EM RB. NO RETORNO, RA E RB ESTAO ACRESCIDOS DO VALOR DE N.

29) PROCEDURE MVW(WORD VALUE N);
CODE

MOVE A SEQUENCIA DE N PALAVRAS, LOCALIZADA A PARTIR DA DE ENDERECO DADO EM RA, PARA A DE ENDERECO DADO EM RB. NO RETORNO, RA E RB ESTAO ACRESCIDOS DO VALOR DE N.

30) PROCEDURE EXEC(...);
CODE

ROTINA DE USO GERAL PARA INTERACAO ENTRE O PROGRAMA E O DOS-HP. ACEITA-SE QUALQUER CHAMADA SINTATICAMENTE VALIDA A ESSA ROTINA, NAO SENDO VERIFICADA A COMPATIBILIDADE DE PARAMETROS.

APENDICE II

LIGACAO COM ROTINAS EM LINGUAGEM SIMBOLICA

UMA INSTRUCAO DE CHAMADA, EM LCS, DA FORMA

```
PROC(P1,P2,...,PN)
```

EQUIVALE A CHAMADA EM "ASSEMBLER":

```
JSB PROC
DEF Q1
DEF Q2
...
DEF QN
```

ONDE:

A) SE PI E' UMA CONSTANTE, DEF QI E' O ENDERECO DESSA CONSTANTE.

B) SE PI E' UMA VARIAVEL, DEF QI E' O ENDERECO (DIRETO OU INDIRETO) DESSA VARIAVEL.

C) SE PI E' UM IDENTIFICADOR DE VETOR, DEF QI E' O ENDERECO (DIRETO OU INDIRETO) DO DESCRITOR DO VETOR (VER ABAIXO).

D) SE PI E' UM IDENTIFICADOR DE ROTULO, DEF QI E' O ENDERECO INDIRETO DE UMA PALAVRA QUE CONTEM A POSICAO DO PROGRAMA CORRESPONDENTE A ESSE ROTULO.

E) SE PI E' UM IDENTIFICADOR DE PROCEDIMENTO (INTERNO OU EXTERNO), DEF QI E' O ENDERECO (POSSIVELMENTE INDIRETO) DO PONTO DE ENTRADA DO PROCEDIMENTO.

ROTINAS EM "ASSEMBLER" CHAMADAS PELA LCS PODEM, PARA A RECEPCAO DOS PARAMETROS, ADOTAR A ENTRADA PADRAO PARA ROTINAS EM LCS:

```
ENT PROC
EXT @PRAM
<PALAVRAS DE CONTROLE>
<AREA DOS PARAMETROS FORMAIS>
...
...
JMP PROC,I
```

AS PALAVRAS DE CONTROLE (DA TRANSMISSAO DE PARAMETROS) TEM A SEGUINTE COMPOSICAO:

OS BITS DE 15 A 10 DA PRIMEIRA CONTEM O NUMERO DE PARAMETROS, SEGUINDO-SE 5 PARES DE BITS AINDA NA PRIMEIRA PALAVRA E 8 EM CADA UMA DAS PALAVRAS SUBSEQUENTES, CADA PAR

CONTENDO INFORMACOES SOBRE O PARAMETRO A QUE CORRESPONDE PELA ORDEM.

UM PAR DE BITS E' FORMADO DE ACORDO COM A REGRA SEGUINTE: O PRIMEIRO BIT E' 1 SE O PARAMETRO FOR ESPECIFICADO WORD VALUE OU DOUBLE VALUE, SENDO ZERO EM CASO CONTRARIO; O SEGUNDO BIT SO E' CONSIDERADO SE O PRIMEIRO ESTIVER LIGADO, E SERA 1 SE O PARAMETRO FOR DE TIPO DUPLO.

A AREA DE PARAMETROS FORMAIS DEVE CONTER EXATAMENTE O NUMERO DE PALAVRAS NECESSARIAS: DUAS PARA PARAMETRO ESPECIFICADO DOUBLE VALUE, UMA PARA OS DEMAIS CASOS.

EXEMPLO:

```
PROCEDURE X(WORD(*) VET, WORD VALUE P,
            DOUBLE VALUE Q, LABEL SAIDA);

X      NOP
       JSB @PRAM
       OCT 10260
VET    BSS 1
P      BSS 1
Q      BSS 2
SAIDA  BSS 1
       ...
```

OBSERVACOES:

1) O DESCRITOR DE UM VETOR E' COMPOSTO DE 3 PALAVRAS: A PRIMEIRA CONTEM O ENDERECO DA SEGUNDA; A SEGUNDA, O ENDERECO DA PRIMEIRA POSICAO DO VETOR, E A TERCEIRA, O NUMERO DE PALAVRAS DO VETOR, COM O SINAL TROCADO.

2) A ROTINA @PRAM PERTENCE A BIBLIOTECA DE ROTINAS INTRINSECAS E PRE-DECLARADAS (RLCS).

APENDICE III

USO DO COMPILADOR IMPLEMENTADO

O COMPILADOR LCS RODA SOB CONTROLE DO DOS-HP E GERA CODIGO-OBJETO EM DISCO COMPATIVEL COM O DOS COMPILADORES FORNECIDOS PELO FABRICANTE.

O PROGRAMA-FONTE PODE SER COMPILADO DIRETAMENTE DE DISCO, CARTAO, FITA DE PAPEL OU FITA MAGNETICA. A LISTAGEM PODE SAIR PELA CONSOLE DE OPERACAO DO SISTEMA, IMPRESSORA, FITA DE PAPEL OU FITA MAGNETICA.

O COMPILADOR E' ACIONADO PELA DIRETIVA:

:PR, LCS [,U1 [,U2]] [,99]

ONDE U1 E' A UNIDADE LOGICA DE ENTRADA E U2 A DE LISTAGEM; OS COLCHETES INDICAM ESPECIFICACOES OPCIONAIS. O SIMBOLO 99 INDICA QUE DEVE SER GERADO EM DISCO O RELOCAVEL, CASO NAO OCORRA ERRO DE COMPILACAO.

SE O PROGRAMA-FONTE ESTIVER EM DISCO, A DIRETIVA ACIMA DEVE SER PRECEDIDA DE UMA DIRETIVA :JF (VER MANUAL DOS-HP).

PARA LIGACAO DE PROGRAMAS OU ROTINAS ESCRITAS EM LCS, ATRAVES DO "LOADER", O USUARIO DEVE ESPECIFICAR, ALEM DE SEUS PROPRIOS ARQUIVOS RELOCAVEIS, A BIBLIOTECA RLCS, DE ROTINAS INTRINSECAS E PRE-DECLARADAS.

APENDICE IV

MENSAGENS DE ERRO DO COMPILADOR

1) ADVERTENCIAS:

101. COMPILACAO INTERROMPIDA
102. CARATER INVALIDO
103. COMENTARIO DESLOCADO
104. INICIALIZACAO INSUFICIENTE
105. INICIALIZACAO EXCESSIVA
106. ITERACAO VAZIA
107. FECHA-PARENTESES ACRESCENTADO
108. OPERACAO ESTRANHA
109. CHAVEAMENTO ESTRANHO

2) ERROS:

201. CARTAO DE CONTROLE INVALIDO
202. FIM DE PROGRAMA INESPERADO
203. CONSTANTE NUMERICA INVALIDA
204. CADEIA DE CARACTERES INVALIDA
205. NUMERO DE CANAL INVALIDO
206. INDICE INCOMPATIVEL COM DIMENSAO
207. DIMENSAO INVALIDA
208. ITEM INCOMPATIVEL COM A DECLARACAO
209. INICIALIZACAO INCOMPATIVEL COM O TIPO
210. FATOR DE REPETICAO INVALIDO
211. ESPECIFICADOR INVALIDO
212. SUPERPOSICAO INVALIDA
213. INDICE DE TIPO INVALIDO
214. CADEIA DE MAIS DE UMA PALAVRA EM CONTEXTO INVALIDO
215. INCOMPATIBILIDADE DE TIPO
216. VARIAVEL DE CONTROLE OMITIDA
217. OPERACAO INVALIDA
218. VARIAVEL SIMPLES REFERENCIADA COMO VETOR
219. VETOR REFERENCIADO COMO VARIAVEL SIMPLES
220. CONDICAO SIMPLES INVALIDA
221. CONDICAO COMPOSTA INVALIDA
222. PARTE ELSE NAO ANTECEDIDA DE CONDICAO
223. IDENTIFICADOR ERRONEAMENTE REFERENCIADO
224. IDENTIFICADOR NAO DECLARADO
225. IDENTIFICADOR JA DECLARADO NO CORRENTE BLOCO
226. CHAMADA INCOMPATIVEL COM PROCEDIMENTO
227. DEFINICAO DE ROTULO JA DEFINIDO
228. DEFINICAO DE ROTULO DESLOCADA
229. DECLARACAO DESLOCADA

230. DIGITO BINARIO ESPERADO
231. ERRO DE SINTAXE
232. PROGRAMA INVALIDO
233. PROGRAMA INCOMPATIVEL COM CARTAO LCS
234. ROTULO NAO DEFINIDO
251. PROGRAMA-OBJETO EXCEDE 32K
252. EXCEDIDA A AREA DE TRABALHO EM DISCO
253. EXCEDIDA A TABELA DE SIMBOLOS
254. EXCEDIDA A TABELA DE BLOCOS
255. NUMERO EXCESSIVO DE PROCEDIMENTOS EXTERNOS
256. NUMERO EXCESSIVO DE VARIAVEIS NA LISTA
257. NUMERO EXCESSIVO DE CONDICOES SIMPLES
258. NUMERO EXCESSIVO DE CASOS PENDENTES
259. NUMERO EXCESSIVO DE PARAMETROS FORMAIS

APENDICE V

REFERENCIAS BIBLIOGRAFICAS

1) WIRTH, NIKLAUS: "PL360, A PROGRAMMING LANGUAGE FOR THE 360 COMPUTERS", "JOURNAL OF THE A.C.M.", VOL. 15, NO. 1, JAN 1968.

2) WIRTH, NIKLAUS: "A CONTRIBUTION TO THE DEVELOPMENT OF ALGOL", "COMMUNICATIONS OF THE A.C.M.", VOL. 9, NO. 6, JUN 1966.

3) WULF, W. A.: "BLISS: A LANGUAGE FOR SYSTEMS PROGRAMMING", "COMMUNICATIONS OF THE A.C.M", VOL. 14 NO. 12, DEC 1971.

4) GRIES, DAVID: "COMPILER CONSTRUCTION FOR DIGITAL COMPUTERS", JOHN WILEY & SONS, INC.

5) GRIES, DAVID: "THE USE OF TRANSITION MATRICES IN COMPILING", "COMMUNICATIONS OF THE A.C.M", VOL. 11, NO. 1, JAN 1968.

6) --: "LANGAGE LP15 - MANUEL D'UTILISATION", COMPAGNIE INTERNATIONALE POUR L'INFORMATIQUE, REF. 4109 E1/FR.

7) --: "A POCKET GUIDE TO THE 2100 COMPUTER", HEWLETT-PACKARD.

8) --: "MOVING-HEAD DISC OPERATING SYSTEM", HEWLETT-PACKARD, REF. 02116-91779.

9) --: "HP ASSEMBLER", HEWLETT-PACKARD, REF. 02116-9014.

10) --: "HP-3000 SYSTEMS PROGRAMMING LANGUAGE", HEWLETT-PACKARD, REF. 03000-90002A.

11) KNUTH, D. E.: "THE ART OF COMPUTER PROGRAMMING", VOL. 1, WESLEY PUBLISHING COMPANY, INC.

12) KNUTH, D. E.: "STRUCTURED PROGRAMMING WITH GO TO STATEMENTS", "A.C.M. COMPUTING SURVEYS", VOL. 6, NO. 4, DEC 1974.

13) BACKUS, J. W.: "THE SYNTAX AND SEMANTICS OF THE PROPOSED INTERNATIONAL ALGEBRAIC LANGUAGE OF THE ZURICH ACM-GAMM CONFERENCE", "PROC. INTERNATIONAL CONF. ON INFORMATION PROCESSING", UNESCO, 1959.

APENDICE VI

PROGRAMA DE EXEMPLO

```
?LCS,P,"BANAV"
&
&
&  SISTEMA DE JOGOS PARA DEMONSTRACAO
&  DOS TERMINAIS DE VIDEO DO SERPRO
&
&     BATALHA  NAVAL
&
PROCEDURE BATALHANAVAL;
   BEGIN
&
   COMMENT: VARIAVEIS GLOBAIS AO SISTEMA;
   WORD CAR DEF 630B;
   WORD TELASDISP DEF 631B;
   WORD(64) ESTADO DEF 632B;
   WORD(126) TELA DEF 732B;
&
   COMMENT: ROTINAS EXTERNAS;
&
   PROCEDURE FANTASMA(WORD VALUE TRANSACAO);
      COMMENT: GERA TRANSACAO FANTASMA
         DE CODIGO ESPECIFICADO;
      CODE;
&
   PROCEDURE ALOCACAO(WORD VALUE TERM,JOGO,SITU);
      COMMENT: ALOCA O TERMINAL ESPECIFICADO PARA O
      JOGO INDICADO, NA SITUACAO INDICADA;
      CODE;
&
   PROCEDURE BLINK(WORD(*) AREA,WORD VALUE P,N);
      COMMENT: DEIXA PISCANDO A CADEIA DE COMPRIMENTO
         N SOBRE O VETOR AREA, A PARTIR DO BYTE P;
      CODE;
&
   COMMENT: VARIAVEIS INTERNAS A ROTINA BATALHANAVAL;
&
   LABEL REINICIO;
   WORD(8) SINCR = 8:0;
   WORD TECL DEF ESTADO(0);
   WORD JOGO DEF ESTADO(1);
   WORD SITU DEF ESTADO(2);
   WORD CASO DEF ESTADO(3);
   WORD ERRO DEF ESTADO(4);
   WORD ADVERS DEF ESTADO(5);
```

```
   WORD PRIMO  DEF ESTADO(6);
   WORD SERIE  DEF ESTADO(7);
   WORD TIRO   DEF ESTADO(8);
   WORD NAVIO  DEF ESTADO(9);
   WORD LINHA  DEF ESTADO(10);
   WORD COLUN  DEF ESTADO(11);
   WORD SALDO  DEF ESTADO(12);
   WORD FALTA  DEF ESTADO(13);
   WORD CURSOR DEF ESTADO(14);
   WORD TOTTEC DEF ESTADO(15);
   WORD TOTADV DEF ESTADO(16);
   WORD TEMP   DEF ESTADO(19);
   WORD(3) RESULT DEF ESTADO(21);
   WORD(5) ESQUA  DEF ESTADO(24);
   WORD(5) CRIVADV DEF ESTADO(29);
   WORD(5) CRIVTEC DEF ESTADO(34);
   WORD(21) SALVA  DEF ESTADO(39);
   WORD(21) LIN.1 DEF TELA(0),LIN.2 DEF TELA(21),
      LIN.3 DEF TELA(42), LIN.4 DEF TELA(63),
      LIN.5 DEF TELA(84),MENSA DEF TELA(105);
   WORD(63) TITULO =
     ("                                                  ",
      "           --  BATALHA  NAVAL  --                 ",
      "                                                  ");
   WORD(10) XALGAR = "0/1U2I304J5K6L7M8\9]";
   WORD PROG=164B, OPER=163B, CORR=142B;
   WORD FIMCPO=137B, TRACO=337B;
   WORD REPRES=130B,MAIS=053B,MENOS=055B;
   WORD(21) BATAERRO =
      "APERTE 'APCPO' PARA PROSSEGUIR !                ";
&
   COMMENT: ROTINAS INTERNAS;
&
   PROCEDURE STCHAR(WORD VALUE LIN,COL,ESQ);
      BEGIN
      COMMENT: COLOCA O CARACTER ENVIADO EM RA NA POSICAO
         DADA POR LIN E COL DA ESQUADRA ESQ;
      DOUBLE SVREG;  WORD CHAR,PTR;
      SVREG:=RBA;
      RA:=RA AND 377B; CHAR:=RA;
      RA:=LIN-1 MPY 42 +COL;
      RB:=RA;  RA:=ESQ;
      IF RA=0 THEN RB:=RB+3 ELSE RB:=RB+22;
      RA:=RB/2;  PTR:=RA;
      IF NOT RB0 THEN
         RA:=TELA(PTR) RTL 8 AND 177400B OR CHAR RTL 8
      ELSE RA:=TELA(PTR) AND 177400B OR CHAR;
      TELA(PTR):=RA;
      RBA:=SVREG
      END STCHAR;
&
   PROCEDURE TECLAINVALIDA;
```

```
      BEGIN
      WORD(21) MSG = ("TECLA INVALIDA",14:"  ");
      MOVE(21,MENSA,0,SALVA,0);
      MOVE(21,MSG,0,MENSA,0);
      BLINK(MENSA,0,14);
      RA:=-1; ERRO:=RA
      END TECLAINVALIDA;
&
   PROCEDURE POSICAOINVALIDA(WORD VALUE ND);
      BEGIN
      WORD P;
      WORD(42) MSG = ("POSICAO INVALIDA",13:"  ",
         "POSICAO INCOMPATIVEL COM ANTERIOR ",4:"  ");
      RA:=ND;
      IF RA<0 THEN [RB:=21; P:=RB]
      ELSE [RA:=-RA; ND:=RA; RB:=0; P:=RB];
      IF RA≤0 THEN FOR ND DO
         BEGIN
         RA:=" "; STBYTE(TELA,CURSOR);
         RB:=CURSOR-1; CURSOR:=RB;
         RA:=TRACO; STBYTE(TELA,CURSOR)
         END;
      MOVE(21,MENSA,0,SALVA,0);
      MOVE(21,MSG,P,MENSA,0);
      BLINK(MENSA,0,34);
      RA:=-1; ERRO:=RA
      END POSICAOINVALIDA;
&
   PROCEDURE TRAVA;
      BEGIN
      WORD SALVA;
      SALVA:=RA;
      RA:=-1; ESTADO(62):=RA;
      RA:=SALVA
      END TRAVA;
&
   PROCEDURE DESTRAVA;
      BEGIN
      WORD SALVA;
      SALVA:=RA;
      RA:=0; ESTADO(62):=RA;
      RA:=SALVA
      END DESTRAVA;
&
   PROCEDURE APRESENTA.JOGO;
      BEGIN
      & SITU = 0
      COMMENT:  O JOGO EH APRESENTADO AO USUARIO,
         SOLICITANDO-SE A TELA ADVERSARIA;
      WORD(126) APRESENTACAO =
        ("            **********************            ",
         "            *  BATALHA  NAVAL  *            ",
```

```
      "              ********************              ",
      "                                                ",
      "DIGITE NUMERO DA TELA ADVERSARIA (0 A 7)  ",
      "OU BATA 'PROG' PARA ESCOLHER OUTRO JOGO   ");
&
   MOVE(126,APRESENTACAO,0,TELA,0);
   RA:=TECL; PRIMO:=RA;
   RA:=1; SITU:=RA;
   RA:=0; SINCR(TECL):=RA
   END APRESENTA.JOGO;
&
PROCEDURE FORNECE.ADVERS;
   BEGIN
   & SITU = 1
   COMMENT:  EH PROCESSADA A ESCOLHA DA
      TELA ADVERSARIA;
   WORD(42) TEXTOS =
     ("TELA * NAO DISPONIVEL - REDIGITE          ",
      "AGUARDE RESPOSTA DA TELA *                ");
   SCAN(XALGAR,0,20);
   IF RB>=0 THEN
      BEGIN
      RB:=RB/2; ADVERS:=RB;
      MOVE(63,TITULO,0,TELA,0);
      MOVE(42,TELA,42,TELA,63);
      RB:=-RB+15;
      RA:=TELASDISP RTL RB;
      IF RA>=0 THEN
         OVFL:=1
      ELSE BEGIN
           RA:=ADVERS RTL 8 OR TECL;
           ALOCACAO(RA,JOGO,2)
           END;
      IF OVFL THEN
         BEGIN
         MOVE(21,TEXTOS,0,MENSA,0);
         RA:=ADVERS; DEC(MENSA,5,1)
         END
      ELSE BEGIN
           MOVE(21,TEXTOS,21,MENSA,0);
           RA:=ADVERS; DEC(MENSA,25,1);
           RA:=4; SITU:=RA;
           TRAVA
           END
      END
   ELSE IF RA=PROG THEN
           BEGIN
           RA:=0; JOGO,SITU:=RA;
           RA:=TECL RTL 8 OR PROG;
           FANTASMA(RA)
           END
   END FORNECE.ADVERS;
```

```
&
   PROCEDURE DESAFIA.ADVERS;
      BEGIN
      & SITU = 2
      COMMENT: EH RECEBIDA A TRANSACAO FANTASMA
         QUE ATIVA A SEGUNDA TELA;
      WORD(63) DESAFIO =
        ("VOCE ESTA' DESAFIADO A UMA                    ",
         "                    CONTRA A TELA * !         ",
         "RESPONDA SIM ('S') OU NAO ('N')               ");
      ADVERS,PRIMO:=RA;
      RA:=0; SINCR(TECL):=RA;
      MOVE(21,DESAFIO,0,TELA,0);
      MOVE(63,TITULO,0,TELA,21);
      MOVE(42,DESAFIO,21,TELA,84);
      RA:=ADVERS; DEC(LIN,5,30,1);
      RA:=3; SITU:=RA
      END DESAFIA.ADVERS;
&
   PROCEDURE RESPONDE.DESAFIO;
      BEGIN
      & SITU = 3
      COMMENT:  EXCLUSIVA DA TELA II. PROCESSA
         RESPOSTA AO DESAFIO DA TELA I;
      WORD(63) INSTRUC =
        ("APERTE A BARRA PARA PROSSEGUIR                ",
         "OU 'R' PARA SABER DAS REGRAS DO JOGO          ",
         "BATA 'PROG' PARA ESCOLHER OUTRO JOGO          ");
      MOVE(63,TITULO,0,TELA,0);
      MOVE(21,LIN.1,0,LIN.5,0);
      IF RA="S" THEN
         BEGIN
         RA:=ADVERS RTL 8 OR 201B;
         FANTASMA(RA);
         MOVE(42,INSTRUC,0,TELA,84);
         RA:=5; SITU:=RA
         END
      ELSE IF RA="N" THEN
               BEGIN
               RA:=ADVERS RTL 8 OR 200B;
               FANTASMA(RA);
               MOVE(21,LIN.4,0,LIN.5,0);
               MOVE(21,INSTRUC,42,MENSA,0);
               RA:=0; JOGO,SITU:=RA
               END
           ELSE TECLAINVALIDA
      END RESPONDE.DESAFIO;
&
   PROCEDURE AGUARDA.CONFIRM;
      BEGIN
      & SITU = 4
      COMMENT: EXCLUSIVO DA PRIMEIRA TELA. PROCESSA
```

```
      RESPOSTA DA SEGUNDA TELA AO DESAFIO;
   WORD(84) INSTRUC =
     ("APERTE A BARRA PARA PROSSEGUIR              ",
      "OU 'R' PARA SABER DAS REGRAS DO JOGO        ",
      "TELA * RECUSOU JOGAR - INDIQUE OUTRA        ",
      "OU BATA 'PROG' PARA ESCOLHER OUTRO JOGO     ");
   IF RA=201B THEN
      BEGIN
      MOVE(42,INSTRUC,0,TELA,84);
      RA:=5; SITU:=RA;
      DESTRAVA
      END
   ELSE IF RA=200B THEN
           BEGIN
           MOVE(42,INSTRUC,42,TELA,84);
           RA:=ADVERS; DEC(LIN,5,5,1);
           RA:=1; SITU:=RA;
           DESTRAVA
           END
   END AGUARDA.CONFIRM;
&
PROCEDURE APRESENTA.REGRAS;
   BEGIN
   & SITU = 5
   COMMENT:  APRESENTA, SE DIGITADA A LETRA 'R',
      AS REGRAS DO JOGO. AO FINAL, SOLICITA
      POSICIONAMENTO DA ESQUADRA;
   WORD(21) BATABARRA =
      "APERTE A BARRA PARA PROSSEGUIR              ";
   WORD(126) QUADRO =
     ("T*  ..............    ..............  T*",
      "T*  ..............    ..............  T*",
      "T*  ..............    ..............  T*",
      "T*  ..............    ..............  T*",
      "T*  ..............    ..............  T*",
      "POSICAO DAS FRAGATAS:                    ");
   WORD(756) REGRAS =
     ("  O JOGO SIMULA UMA BATALHA ENTRE DUAS ES-",
      "QUADRAS, CADA UMA COMPOSTA DE  3  FRAGATAS",
      "E  4  SUBMARINOS, REPRESENTADOS EM UM CAM-",
      "PO RETANGULAR DE 5 LINHAS (A,B,C,D,E)  POR",
      "15 COLUNAS (NUMERADAS DE 1 A 15).          ",
      "  COMPLETADO O POSICIONAMENTO DAS DUAS ES-",
      "QUADRAS,  CABE A CADA JOGADOR,  ALTERNADA-",
      "MENTE, UMA SALVA DE 3 TIROS, ATE' QUE SEJA",
      "TOTALMENTE ELIMINADA UMA DAS FORMACOES, OU",
      "QUE ALGUMA DESISTA DA BATALHA.             ",
      "  CADA JOGADOR ACOMPANHA A DISPUTA ATRAVES",
      "DE UM MAPA ONDE A ESQUERDA E' REPRESENTADA",
      "SUA FROTA E ASSINALADOS OS TIROS DO ADVER-",
      "SARIO  E  A DIREITA SAO INDICADOS OS TIROS",
      "DISPARADOS SOBRE A FORMACAO INIMIGA.       ",
```

```
      "  UMA FRAGATA E' REPRESENTADA:    XX        ",
      "  UM SUBMARINO E' REPRESENTADO:    X        ",
      "  ENTRE DOIS NAVIOS COLOCADOS EM UMA MESMA",
      "LINHA DEVERA' HAVER  PELO MENOS  UM ESPACO",
      "EM BRANCO.                                  ",
      "  OS TIROS CERTEIROS SAO ASSINALADOS '+' E",
      "OS TIROS ERRADOS SAO INDICADOS '-'.         ",
      "  O RESULTADO DE UMA SALVA SO' E' INFORMA-",
      "DO A ESQUADRA QUE ATIRA APOS A ESPECIFICA-",
      "CAO DO TERCEIRO E ULTIMO TIRO.              ",
      "  A POSICAO DE UM NAVIO OU DE UM TIRO  E' ",
      "INDICADA NA FORMA:  L## , ONDE:             ",
      "1. L REPRESENTA A LINHA (DE A A E)          ",
      "2. ## REPRESENTA A COLUNA (1 A 15 , EM 2  ",
      "   ALGARISMOS OU ALGARISMO-FIM DE CAMPO). ",
      "POSICIONE SUA ESQUADRA                      ",
      "         E                                  ",
      "   BOA  PONTARIA !!                         ",
      "                                            ",
      "APERTE BARRA PARA PROSSEGUIR                ",
      "OU 'R' PARA REVER AS REGRAS DO JOGO         ");
&
   PROCEDURE CONFIGURA;
      BEGIN
      WORD CONT;
      MOVE(126,QUADRO,0,TELA,0);
      RA:=-5; CONT:=RA;
      FOR CONT DO
         BEGIN
         RA:=CONT+5 MPY 42; RB:=RA+1;
         RA:=TECL; DEC(TELA,RB,1);
         RB:=RB+40;
         RA:=ADVERS; DEC(TELA,RB,1)
         END;
      RA:=232; CURSOR:=RA;
      RA:=TRACO; STBYTE(TELA,CURSOR);
      RA:=0; CASO:=RA;
      RA:=6; SITU:=RA
      END CONFIGURA;
&
   RB:=CASO;
   IF RB=0 THEN
      IF RA="R" THEN
         BEGIN
         MOVE(105,REGRAS,0,TELA,0);
         MOVE(21,BATABARRA,0,MENSA,0);
         INCR CASO
         END
      ELSE [ IF RA=" " OR RA="/" THEN CONFIGURA ]
   ELSE IF RA=" " OR RA="/" THEN
           BEGIN
           RA:=CASO MPY 105;
```

```
            IF RA=630 THEN
               [RB:=0; CASO:=RB; RB:=126]
            ELSE [INCR CASO; RB:=105];
            MOVE(RB,REGRAS,RA,TELA,0)
            END
   END APRESENTA.REGRAS;
&
   PROCEDURE LETRA;
      BEGIN
      COMMENT:  PROCESSA A REPRESENTACAO
         DE UMA LINHA ('A' A 'E');
      RA:=CAR;
      IF RA>="A" AND RA<="Z" THEN
         IF RA<="E" THEN
            BEGIN
            STBYTE(TELA,CURSOR);
            RA:=RA-100B; LINHA:=RA;
            INCR CURSOR;
            RA:=TRACO; STBYTE(TELA,CURSOR);
            INCR CASO
            END
         ELSE POSICAOINVALIDA(0)
      ELSE TECLAINVALIDA
      END LETRA;
&
   PROCEDURE ALGAR.1;
      BEGIN
      COMMENT:  PROCESSA O PRIMEIRO ALGARISMO DA
         REPRESENTACAO DE UMA COLUNA;
      SCAN(XALGAR,0,20);
      IF RB>=0 THEN
         BEGIN
         RB:=RB/2; COLUN:=RB;
         RA:=RB OR 60B; STBYTE(TELA,CURSOR);
         INCR CURSOR;
         RA:=TRACO; STBYTE(TELA,CURSOR);
         INCR CASO
         END
      ELSE TECLAINVALIDA
      END ALGAR.1;
&
   PROCEDURE ALGAR.2;
      BEGIN
      COMMENT:  PROCESSA O SEGUNDO ALGARISMO DA
         REPRESENTACAO DE UMA COLUNA;
      SCAN(XALGAR,0,20);
      IF RB>=0 THEN
         BEGIN
         RB:=RB/2; TEMP:=RB;
         RA:=COLUN MPY 10 + TEMP;
         IF RA=0 OR RA>15 THEN
            BEGIN
```

```
            POSICAOINVALIDA(2);
            RA:=CASO-2; CASO:=RA
            END
         ELSE BEGIN
               COLUN:=RA;
               RA:=TEMP OR 60B; STBYTE(TELA,CURSOR);
               RB:=CURSOR+2; CURSOR:=RB;
               RA:=TRACO; STBYTE(TELA,CURSOR);
               INCR CASO;
               GOTO REINICIO
               END
         END
      ELSE IF RA=FIMCPO THEN
               BEGIN
               RA:=COLUN;
               IF RA=0 THEN
                  BEGIN
                  POSICAOINVALIDA(2);
                  RA:=CURSOR-2; CURSOR:=RA
                  END
               ELSE BEGIN
                     RB:=CURSOR-1;
                     LDBYTE(TELA,RB);
                     STBYTE(TELA,CURSOR);
                     RA:=60B; STBYTE(TELA,RB);
                     RB:=CURSOR+2; CURSOR:=RB;
                     RA:=TRACO; STBYTE(TELA,CURSOR);
                     INCR CASO;
                     GOTO REINICIO
                     END
               END
           ELSE TECLAINVALIDA
      END ALGAR.2;
&
   PROCEDURE TF.POSIC;
      BEGIN
      COMMENT:  PROCESSA COORDENADAS REFERENTES
         A POSICIONAMENTO;
      LABEL SAIDA;
      WORD APONT;
      WORD(42) DIZERES =
        ("POSICAO DOS SUBMARINOS:                          ",
         "AGUARDE COMPLETAR-SE FORMACAO ADVERSARIA  ");
      RA:=NAVIO-3;
      IF RA≤0 THEN
         BEGIN
         & FRAGATA
         RA:=COLUN;
         IF RA=15 THEN
            BEGIN
            POSICAOINVALIDA(4);
            RA:=0; CASO:=RA;
```

```
        GOTO SAIDA
        END;
    RA:=LINHA-1 MPY 16 + COLUN-1;
    APONT:=RA; RA:=-4;
    FOR RA DO
        BEGIN
        RB:=APONT+RA+3; BIT(ESQUA,RB);
        IF EXT THEN
            BEGIN
            POSICAOINVALIDA(-4);
            RA:=0; CASO:=RA;
            GOTO SAIDA
            END
        END;
    SET(ESQUA,APONT);
    INCR APONT;
    SET(ESQUA,APONT);
    RA:=REPRES;
    STCHAR(LINHA,COLUN,0);
    INCR COLUN;
    STCHAR(LINHA,COLUN,0);
    RA:=NAVIO+1; NAVIO:=RA;
    IF RA=3 THEN
        BEGIN
        MOVE(21,DIZERES,0,MENSA,0);
        RA:=234; CURSOR:=RA;
        RA:=TRACO; STBYTE(TELA,CURSOR);
        END;
    RA:=0; CASO:=RA;
    END FRAGATA
ELSE
    BEGIN
    & SUBMARINO
    RA:=LINHA-1 MPY 16 + COLUN-1;
    APONT:=RA; RA:=-3;
    FOR RA DO
        BEGIN
        RB:=APONT+RA+2; BIT(ESQUA,RB);
        IF EXT THEN
            BEGIN
            POSICAOINVALIDA(-4);
            RA:=0; CASO:=RA;
            GOTO SAIDA
            END
        END;
    SET(ESQUA,APONT);
    RA:=REPRES;
    STCHAR(LINHA,COLUN,0);
    RA:=NAVIO+1; NAVIO:=RA;
    IF RA#7 THEN
        BEGIN
        RA:=0; CASO:=RA
```

```
            END
         ELSE BEGIN
              RA:=-1; SINCR(TECL):=RA;
              RA:=10;
               SALDO,FALTA:=RA;
              RA:=TECL;
              IF RA=PRIMO THEN RA:=4 ELSE RA:=5;
              CASO:=RA;
              RB:=SINCR(ADVERS);
              IF RB<0 THEN
                 BEGIN
                 RA:=ADVERS RTL 8;
                 FANTASMA(RA);
                 GOTO REINICIO
                 END
              ELSE BEGIN
                   MOVE(21,DIZERES,21,MENSA,0);
                   TRAVA
                   END
              END
         END SUBMARINO;
      SAIDA:
      END TF.POSIC;
&
   PROCEDURE ANUNCIA.ENVIO;
      BEGIN
      COMMENT:  APRESENTA A OPCAO DE PROSSEGUIMENTO
         OU DESISTENCIA;
      WORD(21) DECISAO =
         "BARRA PARA ATIRAR OU 'OPER' PARA DESISTIR";
      MOVE(21,DECISAO,0,MENSA,0);
      DESTRAVA
      END ANUNCIA.ENVIO;
&
   PROCEDURE ANUNCIA.RECEPCAO;
      BEGIN
      COMMENT:  ANUNCIA INICIATIVA DO
         ADVERSARIO;
      WORD(21) ESPERA =
         "COM A INICIATIVA O ADVERSARIO    T* X T*  ";
      MOVE(21,ESPERA,0,MENSA,0);
      RA:=TECL OR 60B;   STBYTE(MENSA,34);
      RA:=ADVERS OR 60B; STBYTE(MENSA,39);
      BLINK(MENSA,38,2);
      TRAVA
      END ANUNCIA.RECEPCAO;
&
   PROCEDURE SINCRON.I;
      BEGIN
      COMMENT:  EXCLUSIVO DA TELA I,
         AGUARDA POSICIONAMENTO DA TELA II;
      ANUNCIA.ENVIO;
```

```
      RA:=0; CASO:=RA;
      RA:=7; SITU:=RA;
      END SINCRON.I;
&
   PROCEDURE SINCRON.II;
      BEGIN
      COMMENT:  EXCLUSIVO DA TELA II.
         AGUARDA POSICIONAMENTO DA TELA I;
      ANUNCIA.RECEPCAO;
      RA:=0; CASO:=RA;
      RA:=9; SITU:=RA;
      END SINCRON.II;
&
   PROCEDURE POSICIONA.ESQUADRA;
      BEGIN
      & SITU = 6
      COMMENT:  PROCESSA O POSICIONAMENTO
         DAS ESQUADRAS;
      CASE CASO
      BEGIN
         LETRA;
         ALGAR.1;
         ALGAR.2;
         TF.POSIC;
         SINCRON.I;
         SINCRON.II
      END
      END POSICIONA.ESQUADRA;
&
   PROCEDURE DECIDE.PROSSEG;
      BEGIN
      & SITU = 7
      COMMENT:  PROCESSA INDICACAO PARA
         PROSSEGUIMENTO OU DESISTENCIA;
      WORD(21) SIGA =
         "SERIE **:                                T* X T*   ";
      IF RA=OPER THEN
         BEGIN
         & PERDEU POR DESISTENCIA
         RA:=ADVERS RTL 8 OR 2778;
         FANTASMA(RA);
         RA:=10; SITU:=RA;
         RA:=202B; CAR:=RA;
         GOTO REINICIO
         END
      ELSE IF RA=" " OR RA="/" THEN
               BEGIN
               MOVE(21,SIGA,0,MENSA,0);
               RA:=SERIE+1; SERIE:=RA;
               DEC(MENSA,6,2);
               RA:=0; TIRO:=RA;
               RA:=221; CURSOR:=RA;
```

```
                RA:=TRACO; STBYTE(TELA,CURSOR);
                RA:=TECL OR 60B; STBYTE(MENSA,34);
                RA:=ADVERS OR 60B; STBYTE(MENSA,39);
                BLINK(MENSA,33,2);
                RA:=8; SITU:=RA
                END
        END DECIDE.PROSSEG;
&
    PROCEDURE TF.TIRO;
        BEGIN
        COMMENT:  PROCESSA COORDENADAS
            REFERENTES A UM TIRO;
        RA:=LINHA RTL 4 OR COLUN; RB:=RA;
        RA:=ADVERS RTL 8 OR RB;
        FANTASMA(RA);
        INCR CASO,TIRO;
        TRAVA
        END TF.TIRO;
&
    PROCEDURE ECO.TIRO;
        BEGIN
        COMMENT:  PROCESSA TRANSACAO FANTASMA
            QUE INDICA RECEPCAO DE TIRO;
        WORD APONT;
        INCR TOTTEC;
        RA:=RA RTL 4 OR LINHA RTL 4 OR COLUN;
        RB:=TIRO-1; RESULT(RB):=RA;
        RA:=LINHA-1 MPY 16 + COLUN-1;
        APONT:=RA;
        RA:=CAR; RB:=TIRO;
        IF RA=MAIS THEN
            BEGIN
            BIT(CRIVADV,APONT);
            IF NOT EXT THEN
                BEGIN
                SET(CRIVADV,APONT);
                RA:=FALTA-1; FALTA:=RA;
                IF RA=0 THEN RB:=3
                END
            END;
        IF RB=3 THEN
            BEGIN
            & MOSTRA RESULTADO
            RA:=-TIRO; TEMP:=RA;
            FOR TEMP DO
                BEGIN
                RB:=TEMP+TIRO;
                RA:=RESULT(RB) AND 377B; RB:=0;
                RBA:=RBA RTL 12; LINHA:=RB;
                RA:=RA RTL 4; COLUN:=RA;
                RB:=TEMP+TIRO;
                RA:=RESULT(RB) RTL 8;
```

```
            STCHAR(LINHA,COLUN,1)
            END
         END;
      RA:=FALTA; RB:=TIRO;
      IF RA=0 THEN
         BEGIN
         & VENCEU
         RA:=10; SITU:=RA;
         RA:=201B; CAR:=RA;
         GOTO REINICIO
         END
      ELSE IF RB=3 THEN
               BEGIN
               ANUNCIA.RECEPCAO;
               RA:=ADVERS RTL 8 OR 270B;
               FANTASMA(RA);
               RA:=0; CASO:=RA;
               RA:=9; SITU:=RA;
               END
            ELSE BEGIN
                 DESTRAVA;
                 RA:=0; CASO:=RA
                 END
      END ECO.TIRO;
&
   PROCEDURE MANDA.TIRO;
      BEGIN
      & SITU = 8
      COMMENT:  PROCESSA ESPECIFICACAO
         DOS TIROS;
      CASE CASO
      BEGIN
         LETRA;
         ALGAR.1;
         ALGAR.2;
         TF.TIRO;
         ECO.TIRO
      END
      END MANDA.TIRO;
&
   PROCEDURE ESPERA.TIRO;
      BEGIN
      & SITU = 9
      COMMENT:  PROCESSA TIRO DO ADVERSARIO, DEVOLVENDO
         RESULTADO (DENTRO: "+", FORA: "-").
         DESISTENCIA DO ADVERSARIO EH INDICADA 277B.
         DEVOLUCAO DE CONTROLE EH INDICADA 270B;
      IF RA=270B THEN
         BEGIN
         ANUNCIA.ENVIO;
         RA:=7; SITU:=RA;
         END
```

```
    ELSE IF RA=277B THEN
            BEGIN
            & VENCEU POR DESISTENCIA
            RA:=10; SITU:=RA;
            RA:=203B; CAR:=RA;
            GOTO REINICIO
            END
         ELSE BEGIN
              WORD APONT;
              INCR TOTADV;
              RB:=0;
              RBA:=RBA SHL 12; LINHA:=RB;
              RA:=RA RTL 4; COLUN:=RA;
              RA:=LINHA-1 MPY 16 + COLUN-1;
              APONT:=RA;
              BIT(ESQUA,RA);
              IF EXT THEN
                 BEGIN
                 BIT(CRIVTEC,APONT);
                 IF NOT EXT THEN
                    BEGIN
                    SET(CRIVTEC,APONT);
                    RB:=SALDO-1; SALDO:=RB
                    END;
                 RA:=MAIS
                 END
              ELSE RA:=MENOS;
              STCHAR(LINHA,COLUN,0);
              RB:=RA;
              RA:=ADVERS RTL 8 OR RB;
              FANTASMA(RA);
              RA:=SALDO;
              IF RA=0 THEN
                 BEGIN
                 & PERDEU
                 RA:=10; SITU:=RA;
                 RA:=0; CAR:=RA;
                 GOTO REINICIO
                 END
              END
    END ESPERA.TIRO;
&
  PROCEDURE FORNECE.RESULTADO;
    BEGIN
    & SITU = 10
    COMMENT:  INDICA RESULTADO DO JOGO, OBSERVADA
       A CONFIGURACAO DE BITS DA TRANSACAO;
    WORD(84) RESULTADO =
      ("SUA ESQUADRA FOI DIZIMADA PELO INIMIGO !  ",
       "ESQUADRA INIMIGA TOTALMENTE ANIQUILADA !  ",
       "SUA ESQUADRA RETIROU-SE DA BATALHA !      ",
       "ESQUADRA INIMIGA POSTA EM FUGA !          ");
```

```
      WORD(105) ESTATISTICAS =
        ("                                                    ",
         "CONTAGEM:           T* X T*                        ",
         "TIROS CERTOS...     ** X **                        ",
         "TIROS ERRADOS..     ** X **                        ",
         "BATA 'PROG' PARA INICIAR OUTRO JOGO !              ");
      RA:=RA AND 3B MPY 21;
      MOVE(21,RESULTADO,RA,TELA,0);
      MOVE(105,ESTATISTICAS,0,TELA,21);
      RA:=TECL;   DEC(LIN.3,20,1);
      RA:=ADVERS; DEC(LIN.3,25,1);
      RA:=-FALTA+10;  DEC(LIN.4,19,2);
      RA:=-SALDO+10;  DEC(LIN.4,24,2);
      RA:=TOTTEC+FALTA-10;  DEC(LIN.5,19,2);
      RA:=TOTADV+SALDO-10;  DEC(LIN.5,24,2);
      RA:=0; JOGO,SITU:=RA;
      DESTRAVA
      END FORNECE.RESULTADO;
&
&
&     BATALHA NAVAL
&     ------------
&
   REINICIO:
   RA:=CAR; RB:=ERRO;
   IF RB#0 THEN
      IF RA=CORR THEN
         BEGIN
         MOVE(21,SALVA,0,MENSA,0);
         RA:=0; ERRO:=RA
         END
      ELSE MOVE(21,BATAERRO,0,MENSA,0)
   ELSE CASE SITU
        BEGIN
           APRESENTA.JOGO;
           FORNECE.ADVERS;
           DESAFIA.ADVERS;
           RESPONDE.DESAFIO;
           AGUARDA.CONFIRM;
           APRESENTA.REGRAS;
           POSICIONA.ESQUADRA;
           DECIDE.PROSSEG;
           MANDA.TIRO;
           ESPERA.TIRO;
           FORNECE.RESULTADO
        END
   END BATALHANAVAL;
```