SISTEMA "COPPE-FORTRAN"

UM COMPILADOR FORTRAN RESIDENTE PARA O COMPUTADOR IBM-1130

PEDRO SALENBAUCH

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

APROVADA POR:

DENIS FRANÇA LEITE

CARLOS JOSÉ PEREIRA DE LUCENA

TAMIO SHIMIZU

GUILHERME CHAGAS RODRIGUES

RIO DE JANEIRO
ESTADO DA GUANABARA - BRASIL
SETEMBRO DE 1972

SISTEMA "COPPE-FORTRAN"

UM COMPILADOR FORTRAN RESIDENTE PARA O COMPUTADOR IBM-1130

A PAUL E GRETL

AGRADECIMENTOS
==========

AO PROFESSOR DENIS FRANÇA LEITE, NOSSO ORIENTADOR, PELO CONSTANTE APOIO PROPORCIONADO DURANTE TODO O TRABALHO.

A JOSE CARLOS VIDA CURA E NELSON SIMAS COSTA, PELO AUXILIO PRESTADO NA PROGRAMAÇÃO DO SISTEMA.

A GUILHERME CHAGAS RODRIGUES, JAYME LUIZ SZWARCFITER, LUIZ ANTONIO C. DA C. COUCEIRO, MIGUEL ARANHA BORGES E PAULO MARIO BIANCHI FRANÇA PELAS SUGESTÕES E IDÉIAS PROPORCIONADAS.

A EQUIPE DE PERFURADORES DO NÚCLEO DE COMPUTAÇÃO ELETRONICA (NCE), REPRESENTADA PELO SEU CHEFE, AUGUSTO ANTONIO BARBOSA, PELO TRABALHO DE ENTENDER NOSSA LETRA E PERFURAR AS DEZENAS DE MILHARES DE CARTOES QUE FORMAM O SISTEMA.

A TODOS OS FUNCIONARIOS DO NCE, PELO SUPORTE PROPORCIONADO, BÁSICO PARA A ELABORAÇÃO DO SISTEMA.

RESUMO

E' APRESENTADO O PROBLEMA DA SOBRECARGA DOS CENTROS DE COMPUTAÇÃO DE UNIVERSIDADES, CAUSADO PELA QUANTIDADE ENORME DE NOVOS USUÁRIOS QUE SURGIRAM COM O ENSINO DO FORTRAN AOS ALUNOS. O SISTEMA "COPPE-FORTRAN", UM COMPILADOR FORTRAN RESIDENTE PARA O COMPUTADOR IBM-1130 E' APRESENTADO COMO SOLUÇÃO. ESTE SISTEMA E' DESCRITO EM SEUS VARIOS ASPECTOS, COMO OS OBJETIVOS, COMPONENTES, TÉCNICAS DE IMPLEMENTAÇÃO E OS RESULTADOS OBTIDOS.

THE OVERLOAD OF THE UNIVERSITIES' COMPUTING CENTERS, DUE TO THE LARGE NUMBER OF NEW USERS THAT APPEARED WITH THE FORTRAN TEACHING, IS PRESENTED. THE "COPPE-FORTRAN" SYSTEM, A RESIDENTE LOAD AND GO FORTRAN COMPILER FOR THE IBM-1130 COMPUTER IS INTRODUCED AS A SOLUTION. VARIOUS ASPECTS OF THIS SYSTEM, AS ITS OBJECTIVES, COMPONENTS, IMPLEMENTATION TECHNIQUES AND RESULTS ARE DISCUSSED.

CONTEUDO
========

ÍNDICE DE FIGURAS

CAPITULO I * INTRODUÇÃO: OBJETIVOS *

EXAMINANDO OS PROGRAMAS QUE SAO PROCESSADOS POR UM CENTRO DE COMPUTAÇÃO UNIVERSITÁRIO, VERIFICAMOS QUE BOA PARTE CONSISTE DE PROGRAMAS DE ALUNOS DOS CURSOS INTRODUTÓRIOS DE PROGRAMAÇÃO OU DOS CURSOS QUE UTILIZAM APLICAÇÕES SIMPLES DO COMPUTADOR. RELAÇÕES TÍPICAS EM NÚMERO, QUE ENCONTRAMOS SÃO DE 80% DE PROGRAMAS DE ALUNOS PARA 20% DE OUTROS PROGRAMAS. PODEMOS EXPLICAR ISTO NOTANDO QUE UM DOS PRINCIPAIS OBJETIVOS DOS CENTROS DE COMPUTAÇÃO UNIVERSITÁRIOS E' JUSTAMENTE O ENSINO DE LINGUAGENS DE PROGRAMAÇÃO AOS ALUNOS DE SUA UNIVERSIDADE.

ESTES PROGRAMAS, EMBORA PEQUENOS E SIMPLES, CRIAM UMA CARGA CONSIDERÁVEL PARA O COMPUTADOR, DEVIDO A SUA GRANDE QUANTIDADE. DEVIDO AO FATO DE QUE A VELOCIDADE DO DESENVOLVIMENTO TECNOLÓGICO E' MUITO GRANDE, NAO COMPENSA UTILIZARMOS COMPUTADORES SUPER-DIMENSIONADOS, DE MODO QUE EM GERAL A SATURAÇÃO DE UM COMPUTADOR UNIVERSITÁRIO JÁ E' PREVISTA DURANTE A FASE DE AQUISIÇÃO. AO SER ATINGIDO ESTE PONTO, ANTES DE EMPREENDERMOS A AQUISIÇÃO DE EQUIPAMENTO DE MAIOR PORTE, DEVEMOS TENTAR OTIMIZAR A UTILIZAÇÃO DO EQUIPAMENTO DISPONÍVEL.

ESTA OTIMIZAÇÃO DEVE ATINGIR NATURALMENTE OS SERVIÇOS QUE PRODUZEM OS MAIORES GASTOS, QUE NO CASO DE CENTROS DE COMPUTAÇÃO UNIVERSITÁRIOS SAO REPRESENTADOS PELOS PROGRAMAS DE ALUNOS, COMO JÁ FOI CITADO. ESTA OTIMIZAÇÃO E', PELO MENOS APARENTEMENTE, POSSÍVEL, POIS COMO SE TRATA DE UM GRANDE NÚMERO DE PROGRAMAS, PODEMOS PELO MENOS MINIMIZAR O TEMPO DE TRANSIÇÃO ENTRE O PROGRAMA DE UM ALUNO E O DE OUTRO.

ESTES PROBLEMAS ESTAVAM SENDO SENTIDOS PELO ENTÃO "DEPARTAMENTO DE CÁLCULO CIENTÍFICO" DA COPPE, EM MEADOS DE 1968. O DEPARTAMENTO POSSUÍA UM COMPUTADOR IBM-1130, QUE COM SEUS 250 USUÁRIOS DA ÉPOCA ESTAVA CAMINHANDO RAPIDAMENTE PARA A SATURAÇÃO. JÁ NAQUELA EPOCA ERAM MINISTRADOS CURSOS DE FORTRAN PARA OS ALUNOS, SENDO QUE UM DOS REQUISITOS PARA A CONCLUSÃO DO CURSO ERA A APRESENTAÇÃO DE UM PROGRAMA REALIZADO PELO ALUNO. EM MÉDIA, 80 ALUNOS CONCLUÍAM ESTE CURSO MENSALMENTE, CADA UM USANDO VÁRIAS VEZES O COMPUTADOR.

A IDÉIA DE OTIMIZAR SISTEMAS OPERACIONAIS DE COMPUTADORES PARA AUMENTAR SUA EFICIÊNCIA NO PROCESSAMENTO DE PROGRAMAS ORIENTADOS PARA CERTO TIPO DE APLICAÇÃO NAO E' NOVA; VÁRIAS UNIVERSIDADES NORTE-AMERICANAS JÁ DESENVOLVERAM TRABALHOS NESTE SENTIDO, E OS RESULTADOS FORAM COMPENSADORES. DEVE-SE RESSALTAR NO ENTANTO, QUE ESTAS EXPERIÊNCIAS FORAM EM GERAL REALIZADAS EM SISTEMAS DE GRANDE PORTE, QUE NAO E' O NOSSO CASO.

A EQUIPE DO DCC CONSIDEROU INTERESSANTE O DESENVOLVIMENTO DE UM SISTEMA ORIENTADO PARA O PROCESSAMENTO DE PROGRAMAS DE ALUNOS, MAS DEVIDO A FALTA DE RECURSOS E A MAIOR PRIORIDADE DE OUTROS EMPREENDIMENTOS, A IDÉIA FOI TEMPORARIAMENTE POSTA DE LADO.

SOMENTE EM 1970 A IDÉIA FOI RETOMADA, QUANDO COINCIDIRAM OS INTERESSES DO DCC EM VÊ-LA DESENVOLVIDA, E A DESTE AUTOR EM REALIZÁ-LA COMO TESE.

O OBJETIVO DO TRABALHO INCLUÍA NAO SOMENTE O ESTUDO DA VIABILIDADE E DAS TÉCNICAS QUE SERÃO EMPREGADAS, MAS TAMBEM A OBTENÇÃO DE UM PRODUTO FINAL QUE FUNCIONE, AFIM DE QUE AS IDÉIAS POSSAM SER VERIFICADAS NA PRÁTICA. CONSIDERANDO QUE O COMPUTADOR IBM-1130 E' BASTANTE DIFUNDIDO ENTRE AS UNIVERSIDADES BRASILEIRAS, ESTE PRODUTO PODE TAMBEM SER DE GRANDE VALIA PARA ESTAS UNIVERSIDADES.

```
*****************************************
*                                       *
* CAPITULO II   * CONSIDERAÇÕES INICIAIS*
*                                       *
*****************************************
```

COMO JÁ DISSEMOS ANTERIORMENTE, TENTAREMOS IDEALIZAR UM SISTEMA OTIMIZADO PARA O PROCESSAMENTO DE PROGRAMAS PEQUENOS E SIMPLES, TAIS COMO OS ENCONTRADOS NOS CENTROS DE COMPUTAÇÃO UNIVERSITÁRIOS. PARA TANTO, IREMOS ANALISAR O "MONITOR" - O SISTEMA OPERACIONAL FORNECIDO PELA IBM PARA O COMPUTADOR 1130, E DETETAR PONTOS, QUE DE ACORDO COM O NOSSO TIPO DE CARGA, PERMITEM OTIMIZAÇÕES.

PELA OBSERVAÇÃO DESTA CARGA, CONSTATAMOS QUE OS CITADOS PROGRAMAS TEM AS SEGUINTES CARACTERÍSTICAS:

1. OS PROGRAMAS POSSUEM MUITOS ERROS DEVIDO AO FATO DE SEREM ESCRITOS POR PROGRAMADORES PRINCIPIANTES. EM CONSEQUÊNCIA, CADA PROGRAMA E' COMPILADO MUITAS VEZES, TANTO PELOS ERROS QUANTO PORQUE OS PROGRAMADORES NAO TEM PRÁTICA DE DEPURAÇÃO.

2. OS PROGRAMAS SAO EM GERAL CURTOS, CONTENDO EM MEDIA APENAS 40 CARTÕES, COMPARADOS AOS 300 OU MAIS QUE UM PROGRAMA UM POUCO MAIS COMPLEXO GERALMENTE TEM.

3. O NUMERO DE PROGRAMAS PROCESSADOS E' MUITO GRANDE, DA ORDEM DE CENTENAS OU ATÉ MILHARES POR DIA.

4. CADA PROGRAMA UMA VEZ CORRIGIDO, EM GERAL SO E' EXECUTADO UMA VEZ, POIS DEPOIS DE OBTIDO O RESULTADO DESEJADO, NAO APRESENTA MAIS INTERESSE.

5. EM GERAL, ESTES PROGRAMAS UTILIZAM POUCOS RECURSOS DO COMPUTADOR, COMO POR EXEMPLO, POUCA MEMORIA, APENAS SUBROTINAS E FUNÇÕES PADROES, E SOMENTE EQUIPAMENTOS PERIFÉRICOS CONVENCIONAIS, TAIS COMO A LEITORA DE CARTÕES E A IMPRESSORA.

NO PASSO SEGUINTE, IREMOS VERIFICAR COMO E' QUE O MONITOR SE COMPORTA COM A CARGA DESCRITA ACIMA, COM O INTUITO DE VERIFICAR AS POSSIBILIDADES DE OTIMIZAÇÃO. PODEMOS PERCEBER AS SEGUINTES CARACTERÍSTICAS MAIS IMPORTANTES:

1. O COMPILADOR E' LONGO E COMPLEXO, POIS PROVÊ UMA SÉRIE DE RECURSOS NAO UTILIZADOS OU NECESSÁRIOS PELOS PROGRAMAS CITADOS, COMO A OTIMIZAÇÃO DO PROGRAMA OBJETO, E PERIFÉRICOS NÃO CONVENCIONAIS.

2. COMO O FABRICANTE CONDICIONA O COMPILADOR A FUNCIONAR COM A CONFIGURAÇÃO MÍNIMA DE MEMÓRIA, ÊLE TEM DE SER DIVIDIDO EM VÁRIAS FASES, ISTO E', VÁRIOS TRECHOS DO COMPILADOR QUE UTILIZAM A MESMA MEMÓRIA EM MOMENTOS DIFERENTES, QUE SÃO LIDOS DO DISCO.

3. O PROGRAMA OBJETO E' GERADO EM UMA LINGUAGEM INTERMEDIÁRIA, QUE O TORNA IMPROPRIO PARA CARGA E EXECUÇÃO IMEDIATA, MAS QUE AINDA DEVERÁ SER PROCESSADO PELO "CORE LOAD BUILDER", CUJA FUNÇÃO PRINCIPAL E' INCORPORAR AO PROGRAMA OBJETO AS SUBROTINAS E FUNÇÕES UTILIZADAS PELO PROGRAMADOR.

4. DURANTE A EXECUÇÃO DO "CORE LOAD BUILDER", AS ROTINAS NECESSÁRIAS DURANTE A EXECUÇÃO E OS SUBPROGRAMAS REFERENCIADOS PELO PROGRAMA TERÃO DE SER LIDOS DO DISCO.

5. O COMPILADOR FORNECE DIAGNÓSTICOS QUE SÃO SUFICIENTES PARA PROGRAMADORES EXPERIENTES, MAS QUE SAO VAGOS PARA OS PRINCIPIANTES.

6. DURANTE A EXECUÇÃO NAO E' FORNECIDO PRATICAMENTE DIAGNÓSTICO ALGUM, O QUE DIFICULTA A DETEÇÃO DE ERROS DE EXECUÇÃO POR PARTE DO PROGRAMADOR.

7. DURANTE O PROCESSAMENTO DO CARTAO "JOB", ISTO E', DURANTE A TRANSIÇÃO DO PROGRAMA DE UM USUÁRIO E O DE UM OUTRO E' PERDIDO UM TEMPO CONSIDERÁVEL NA ATUALIZAÇAO DE ARQUIVOS DE CONTROLE, RESIDENTES EM DISCO.

TENTAREMOS A SEGUIR, PROPOR SOLUÇÕES, ISTO E', ESTABELECER COMO SERIA UM SISTEMA OTIMIZADO PARA PROCESSAR PROGRAMAS SIMPLES. E' IMPORTANTE SALIENTAR QUE TODOS OS PONTOS LISTADOS ACIMA SAO PASSÍVEIS DE OTIMIZAÇÃO, E QUE PEQUENAS DIMINUIÇÕES NO TEMPO DE PROCESSAMENTO DE CADA PROGRAMA TEM GRANDE SIGNIFICADO DEVIDO A ELEVADA QUANTIDADE DE PROGRAMAS. ESTABELECEMOS AS SEGUINTES PREMISSAS A ATINGIR PARA A OTIMIZAÇÃO:

1. O COMPILADOR NAO REALIZARÁ NENHUMA OTIMIZAÇÃO DO PROGRAMA OBJETO EXCETO AS MAIS ELEMENTARES. NAO TEM SENTIDO PERDER TEMPO OTIMIZANDO UM PROGRAMA QUE POSSIVELMENTE NAO SERÁ EXECUTADO, POIS CONTEM ERROS.

2. ELE SERA INTEIRAMENTE RESIDENTE NA MEMÓRIA E DE APENAS UM PASSO, ISTO E', CONSULTARÁ O PROGRAMA FONTE APENAS UMA VEZ. DESTE MODO OBTEMOS COMPILACOES MAIS RÁPIDAS.

3. O PROGRAMA OBJETO SERÁ ABSOLUTO, GERADO DIRETAMENTE NA MEMÓRIA INTERNA E EXECUTÁVEL SEM NENHUM PÓS-PROCESSAMENTO PELO "CORE LOAD BUILDER" ("LOAD AND GO").

4. SOMENTE SERÁ PERMITIDO AO USUÁRIO O EMPREGO DE SUBPROGRAMAS FORNECIDOS EM LINGUAGEM FORTRAN, DENTRO DO JOB, ALEM DAS SUBROTINAS E FUNÇÕES PADRÕES DA LINGUAGEM. PROIBIMOS ASSIM, O USO DA BIBLIOTECA DE SUBPROGRAMAS, O QUE NAO E' FUNDAMENTAL PARA PRINCIPIANTES. COM ISTO EVITAMOS A FASE DE "LINK-EDIÇÃO", QUE EM GERAL E' BEM DEMORADA.

5. O SISTEMA POSSUIRÁ UM PEQUENO SUPERVISOR, QUE PROCESSARÁ OS CARTÕES DE CONTRÔLE DE CADA PROGRAMA DE UM CONJUNTO DESTES, E SO DEVOLVENDO O CONTROLE AO MONITOR NO FINAL DESTE CONJUNTO. EM VIRTUDE DISTO, TODOS OS PROGRAMAS QUE IRÃO SER PROCESSADOS PELO NOVO SISTEMA DEVERÃO ESTAR REUNIDOS, E PROCESSADOS DE UMA VEZ.

6. TANTO DURANTE A COMPILAÇÃO COMO A EXECUÇÃO SERÃO DADOS DIAGNOSTICOS DETALHADOS, CAPAZES DE PERMITIR AO PRINCIPIANTE CORRIGIR SEUS ERROS SEM TER DE RECORRER A INSTRUTORES.

7. A LINGUAGEM FORTRAN E OS CARTÕES DE CONTROLE UTILIZADOS PELO SISTEMA DEVERÃO SER COMPATÍVEIS COM OS UTILIZADOS PELO MONITOR, PARA QUE NAO HAJA

NECESSIDADE DE APRENDIZAGEM ESPECIAL PARA A UTILIZAÇÃO DO SISTEMA.

EM RESUMO, A IDÉIA FUNDAMENTAL E' A DE CRIARMOS UM SUBSISTEMA DO MONITOR, AUTONOMO E RESIDENTE NA MEMÓRIA, QUE POSSUA A CAPACIDADE DE COMPILAR E EXECUTAR OS PROGRAMAS SIMPLES DE MANEIRA MAIS EFICIENTE POSSIVEL. ESTE SUB-SISTEMA FOI CHAMADO DE "SISTEMA COPPE-FORTRAN", NOME PELO QUAL PASSAREMOS A REFERI-LO.

COMO JÁ FOI MENCIONADO ANTERIORMENTE, A IDÉIA DE UM SISTEMA OTIMIZADO PARA CARGAS ESPECÍFICAS NAO E' NOVA, E DIVERSAS UNIVERSIDADES AMERICANAS ELABORARAM SEUS PRÓPRIOS SISTEMAS. ENTRE ELAS TEMOS:

1. UNIVERSIDADE DE WISCONSIN: A UNIVERSIDADE DE WISCONSIN DESENVOLVEU EM 1961 O "FORGO" PARA O COMPUTADOR IBM-1620 (1). UTILIZA A LINGUAGEM FORTRAN, A TÉCNICA DE "LOAD AND GO", E CONTEM RECURSOS PARA AUXILIAR A DEPURACAO DOS PROGRAMAS.

2. UNIVERSIDADE DE CORNELL: ESTA UNIVERSIDADE CRIOU E IMPLANTOU A LINGUAGEM "CORC" (CORNELL COMPUTING LANGUAGE), EM 1962 (2). FORAM FEITOS COMPILADORES PARA OS COMPUTADORES B-220 E CDC-1604 QUE FORNECIAM EXCELENTES DIAGNOSTICOS DURANTE A EXECUÇÃO. OBTEVE GRANDE SUCESSO NA ÉPOCA, E DE SETEMBRO DE 1962 A ABRIL DE 1963 PROCESSOU MAIS DE 4000 PROGRAMAS, O QUE ERA BEM SIGNIFICATIVO PARA A ÉPOCA.

3. UNIVERSIDADE DE PURDUE: A UNIVERSIDADE DE PURDUE, COM A IMPLANTACAO DO "PUFFT" (PURDUE UNIVERSITY FAST FORTRAN TRANSLATOR), DECIDIU USAR O FORTRAN, AO INVÉS DE CRIAR UMA NOVA LINGUAGEM (3). FOI ELABORADO PARA O COMPUTADOR IBM-7094 E CONSEGUIU VELOCIDADE DE COMPILACAO 10 VEZES SUPERIOR AO DO COMPILADOR "IBFTC", FORNECIDO PELA IBM.

4. UNIVERSIDADE DE WATERLOO: A UNIVERSIDADE DE WATERLOO, EM 1965 REALIZOU O "WATFOR", PARA O IBM-7040/44, ADOTANDO TAMBEM A LINGUAGEM FORTRAN (4,5). TRATA-SE TAMBEM DE UM SISTEMA CONTENDO UM COMPILADOR "LOAD AND GO", QUE POSSUI A INTERESSANTE CARACTERÍSTICA DE PERMITIR TAMBEM A EXECUÇÃO DE PROGRAMAS QUE CONTENHAM ERROS. MAIS TARDE, QUANDO A UNIVERSIDADE RECEBEU UM IBM-360, UM NOVO COMPILADOR FOI FEITO, E RECENTEMENTE FOI LANÇADA UMA VERSÃO MAIS APERFEIÇOADA, O "WATFIV". E' SEM DUVIDA A EXPERIÊNCIA MAIS BEM SUCEDIDA NESTE CAMPO.

****
* *
* CAPITULO III * RECURSOS DISPONÍVEIS *
* *
****

ESTE CAPÍTULO TEM A DUPLA FINALIDADE DE APRESENTAR O COMPUTADOR IBM-1130 AO LEITOR QUE NAO O CONHEÇA, E O DE REALIZAR UM LEVANTAMENTO DOS RECURSOS DISPONÍVEIS, PARA AVALIAR COM O QUE PODEMOS CONTAR. ESTES RECURSOS PODEM SER DIVIDIDOS EM EQUIPAMENTO ("HARDWARE") E SISTEMAS DE PROGRAMAÇÃO ("SOFTWARE").

1. EQUIPAMENTO

O COMPUTADOR IBM-1130 E' UM SISTEMA DE TERCEIRA GERAÇÃO DE PEQUENO PORTE, DIRIGIDO PRINCIPALMENTE PARA USO CIENTÍFICO, EMBORA ATUALMENTE TAMBÉM ESTEJA COMEÇANDO A SER UTILIZADO EM APLICAÇÕES COMERCIAIS. E' UM COMPUTADOR RELATIVAMENTE RÁPIDO (10 MICROSEGUNDOS EM MÉDIA POR INSTRUÇÃO DE MÁQUINA). E' UM SISTEMA MODULAR, QUE PERMITE UMA EXPANSÃO GRADUAL EM TAMANHO DE MEMORIA INTERNA E TIPOS DE PERIFÉRICOS, A MEDIDA QUE AS NECESSIDADES O EXIJAM.

O SISTEMA E' BINÁRIO, ORIENTADO PARA PALAVRAS DE 16 BITS. A MEMÓRIA INTERNA CONTEM NO MÍNIMO 4K PALAVRAS (K = 1024), E PODE SER EXTENDIDA ATÉ 32K. TODAS AS GRANDEZAS ARITMÉTICAS SAO REPRESENTADAS EM FORMA BINÁRIA, E AS INTRUÇÕES DE MÁQUINA PODEM OCUPAR UMA OU DUAS PALAVRAS (16 OU 32 BITS). O TEMPO DE ACESSO A UMA PALAVRA E' DE 3,6 MICROSEGUNDOS (EM ALGUNS MODELOS ESTE VALOR BAIXA PARA 2,2 MICROSEGUNDOS). PARA FACILIDADE DE PROGRAMAÇÃO, SÃO DISPONÍVEIS 3 REGISTROS DE ÍNDICES, CADA QUAL TAMBÉM COM 16 BITS, QUE ESTAO LOCALIZADOS FÍSICAMENTE EM 3 PALAVRAS DA PRÓPRIA MEMORIA.

VÁRIOS PERIFÉRICOS PODEM SER CONECTADOS AO 1130. ENTRE ELES, TEMOS: DISCOS MAGNÉTICOS, 2 MODELOS DE LEITORAS DE CARTÃO, PERFURADORA DE CARTÕES, 2 MODELOS DE IMPRESSORAS, PLOTTER, LEITORA/PERFURADORA DE FITA DE PAPEL, LEITORA DE MARCAS ÓTICAS, DISPLAY, ALEM DO TECLADO E IMPRESSORA CONSOLE

(FIG. 1). DESTES PERIFÉRICOS, VAMOS DECREVER COM MAIS DETALHES OS QUE SAO MAIS COMUMENTE USADOS.

DOS DOIS MODELOS DE LEITORAS DE CARTAO, O MODELO 2501 LE 1000 CARTOES POR MINUTO, E O MODELO 1442 LE 300 OU 400 CARTÕES POR MINUTO, DEPENDENDO DO TIPO (1442-6 OU 1442-7). AS DUAS IMPRESSORAS DISPONÍVEIS SAO: 1403 QUE IMPRIME 210, 340 OU 600 LINHAS POR MINUTO, DEPENDENDO DO TIPO E A 1132, QUE IMPRIME 80 LINHAS POR MINUTO. AMBAS CONTEM UM CONJUNTO DE 48 CARACTERES.

A PERFURADORA DE CARTOES 1442 PODE PERFURAR DE 50 A 300 CARTOES POR MINUTO, DEPENDENDO DO TIPO, E O NÚMERO DE COLUNAS PERFURADAS NO CARTAO.

AS UNIDADES DE DISCO MAGNÉTICO, CUJO NUMERO PODE ALCANCAR UM MAXIMO DE 5 CONTEM DISCOS INTERCAMBIÁVEIS, CUJA CAPACIDADE DE CADA UM E' DE 512000 PALAVRAS. A SUPERFÍCIE DE GRAVAÇÃO DESTES DISCOS E' DIVIDIDA EM 200 CILINDROS, E CADA CILINDRO EM 8 SETORES DE 320 PALAVRAS CADA. O TEMPO MÉDIO DE ACESSO A UM DETERMINADO CILINDRO E' DE 750 MILISEGUNDOS, E A VELOCIDADE DE TRANSFERENCIA E' DE 25600 PALAVRAS POR SEGUNDO.

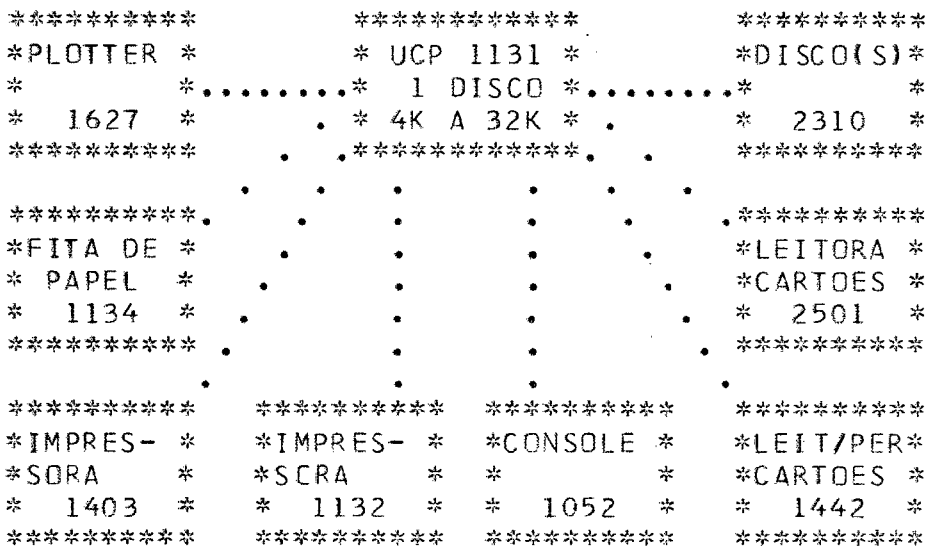


FIG. 1: CONFIGURACAO DO 1130
(O MULTIPLEXOR NAO FOI INDICADO)

O 1130 POSSUI UM SISTEMA DE INTERRUPÇÕES COM 6 NÍVEIS DE PRIORIDADE, E PERMITE A TOTAL SOBREPOSIÇÃO DAS OPERACOES DE ENTRADA/SAIDA COM O PROCESSAMENTO.

PONTOS EM QUE O 1130 POSSIVELMENTE DEIXA A DESEJAR SÃO A AUSÊNCIA DE FITA MAGNÉTICA, A AUSENCIA DE PROTEÇÃO DE MEMÓRIA, A AUSENCIA DE INSTRUÇÕES DE MÁQUINA PARA EFETUAR AS OPERAÇÕES ARITMETICAS DE PONTO FLUTUANTE, E O FATO DE QUE O COMPUTADOR SIMPLESMENTE PARA QUANDO E' TENTADA A EXECUÇÃO DE ALGUMA INSTRUÇÃO INVÁLIDA. PARA MAIORES DETALHES, VER (6).

A CONFIGURAÇÃO DISPONÍVEL PARA ESTE TRABALHO FOI CONSTANTE E E' A SEGUINTE: MEMÓRIA INTERNA DE 32K PALAVRAS, COM TEMPO DE ACESSO DE 3,6 MICROSEGUNDOS, 3 UNIDADES DE DISCO, UMA IMPRESSORA 1403 DE 600 LINHAS POR MINUTO, UMA IMPRESSORA 1132 DE 80 LINHAS POR MINUTO, UMA LEITORA DE CARTOES 2501, DE 1000 CARTOES POR MINUTO, UMA LEITORA/PERFURADORA DE CARTÕES 1442 DE 400 CARTÕES POR MINUTO (NA LEITURA), LEITORA E PERFURADORA DE FITA DE PAPEL, O PLOTTER E O TECLADO/IMPRESSORA CONSOLE.

2. SISTEMAS DE PROGRAMAÇÃO

NA PARTE DE PROGRAMAÇÃO, A IBM FORNECE UM SISTEMA MONITOR, RESIDENTE EM DISCO. ESTE MONITOR PROCESSA "BATCH" DE JOBS SEQUENCIALMENTE, SEM UTILIZAR MULTIPROGRAMAÇÃO. ÊLE E' COMPOSTO DOS SEGUINTES COMPONENTES: UM SUPERVISOR, UM COMPILADOR FORTRAN, UM ASSEMBLER, UM PROGRAMA UTILITARIO PARA O DISCO (DUP), O "CORE LOAD BUILDER", O "CORE IMAGE LOADER", E UMA BIBLIOTECA DE PROGRAMAS/SUBPROGRAMAS (O MONITOR CONTEM TAMBÉM UM COMPILADOR RPG E COBOL LIBERADOS MAIS TARDE, ALÉM DE OUTROS COMPONENTES, QUE NÃO SÃO CONSIDERADOS NESTA DISCUSSÃO). TODOS ESTES COMPONENTES SAO GRAVADOS NO DISCO, DEFINIDO COMO "MESTRE", SEQUENCIALMENTE A PARTIR DO CILINDRO MAIS EXTERNO. O MONITOR COMPLETO OCUPA CÊRCA DE UM QUARTO DO DISCO, EMPREGANDO-SE O RESTANTE DO MESMO COMO AREA DE TRABALHO PARA VARIOS DOS COMPONENTES (FIG. 2). PASSAREMOS A DESCREVER CADA COMPONENTE SUSCINTAMENTE.

O SUPERVISOR E' O RESPONSÁVEL PELAS FUNÇÕES DE CONTROLE DO MONITOR. ELE LE CARTÕES DE CONTROLE PRÓPRIOS, CONTIDOS NO "BATCH", INTERPRETA-OS, CARREGA E PASSA O CONTROLE A OUTRO INTEGRANTE DO SISTEMA PARA EXECUTAR A FUNÇAO DESEJADA.

O COMPILADOR FORTRAN RECEBE PROGRAMAS ESCRITOS NA LINGUAGEM FORTRAN, GERANDO UM PROGRAMA EQUIVALENTE EM LINGUAGEM DE MÁQUINA, NA AREA DE TRABALHO DO DISCO, QUE AINDA NAO ESTA PRONTO PARA A EXECUÇAO; ESTÁ EM UM FORMATO CHAMADO MODULO OBJETO, QUE AINDA DEVERÁ SER PROCESSADO PELO "CORE LOAD BUILDER".

O ASSEMBLER ACEITA PROGRAMAS ESCRITOS EM LINGUAGEM SIMBÓLICA, E MONTA O PROGRAMA OBJETO EM LINGUAGEM DE MAQUINA EQUIVALENTE. SUA SAIDA TAMBEM E' DEIXADA NA ÁREA DE TRABALHO DO DISCO, E NO FORMATO DE MODULO OBJETO. (INICIALMENTE O ASSEMBLER NAO POSSUIA A FACILIDADE DE MACRO-MONTAGEM, QUE SÓ FOI LIBERADA MAIS TARDE).

O PROGRAMA UTILITÁRIO PARA O DISCO (DUP) E' UM PROGRAMA DESTINADO A FACILITAR AO PROGRAMADOR OPERAÇÕES TAIS COMO PERFURAR EM CARTÕES UM ARQUIVO, GERAR ARQUIVOS EM DISCO A PARTIR DE CARTOES, COPIAR UM ARQUIVO DE UM DISCO A OUTRO, ARQUIVAR UM PROGRAMA NA BIBLIOTECA DE PROGRAMAS, ETC...

A FUNCAO DO "CORE LOAD BUILDER" E' A DE RECEBER UM MODULO OBJETO AINDA NÃO EXECUTÁVEL, REUNIR OS DIVERSOS SUBPROGRAMAS NECESSÁRIOS PELO PROGRAMA, RELOCAR CADA UMA DESTAS SUBROTINAS CHAMADAS, DEIXANDO O RESULTADO, UM PROGRAMA EXECUTÁVEL CHAMADO DE MÓDULO DE CARGA, EM OUTRA ÁREA DO DISCO, DE ONDE PODERÁ A SEGUIR SER EXECUTADO OU ARMAZENADO NA BIBLIOTECA.

A UNICA FUNÇÃO DO "CORE IMAGE LOADER" E' A DE CARREGAR UM MÓDULO DE CARGA NA MEMÓRIA INTERNA, E TRANSFERIR O CONTROLE A ÊLE.

FINALMENTE, A BIBLIOTECA DE PROGRAMAS, NO QUAL COMO O NOME INDICA, PODEMOS ARQUIVAR MODULOS OBJETOS, MÓDULOS DE CARGA ALÉM DE ARQUIVOS DE DADOS. A MANUTENÇÃO DESTA BIBLIOTECA E' REALIZADA PELO DUP.

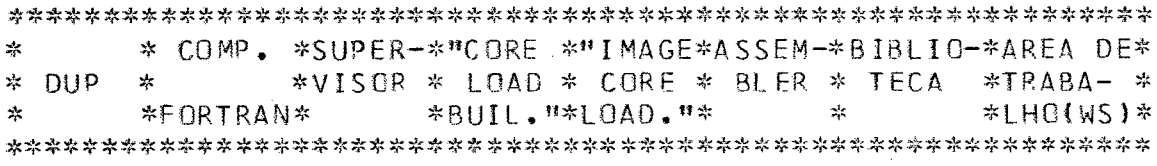


FIG. 2: DISPOSIÇÃO DO MONITOR NO DISCO
(FORA DE ESCALA E CONTENDO
APENAS ALGUNS DOS COMPONENTES)

COMO O MONITOR FOI PROJETADO PARA FUNCIONAR A PARTIR DA MEMÓRIA MÍNIMA, DE 4K, A MAIORIA DOS COMPONENTES ACIMA DESCRITOS SAO DIVIDIDOS EM VÁRIAS FASES, ONDE ESTAS FASES SÃO LIDAS NA MEMÓRIA INTERNA, À MEDIDA QUE SÃO NECESSÁRIOS. COMO EXEMPLO, O COMPILADOR FORTRAN CONTEM 27 FASES, O "CORE LOAD BUILDER" E' FORMADO DE 12 FASES, ETC... ÊSTE ESQUEMA DE FASES E' INDEPENDENTE DO TAMANHO DA MEMÓRIA INTERNA, E PORTANTO O AUMENTO DESTA MEMÓRIA NÃO TRAZ AUMENTO DE VELOCIDADE DE PROCESSAMENTO.

O SISTEMA MONITOR PERMITE A DEFINIÇÃO DE UMA UNIDADE DE ENTRADA E UMA DE SAÍDA DE DADOS COMO SENDO AS UNIDADES PRINCIPAIS. POR INTERMÉDIO DESTAS UNIDADES E' QUE O MONITOR LÊ OS PROGRAMAS A SEREM PROCESSADOS E IMPRIME AS MENSAGENS DESEJADAS. PARA ESTAS UNIDADES NATURALMENTE SAO ESCOLHIDAS AS MAIS RÁPIDAS DISPONÍVEIS. PARA MAIORES DETALHES SOBRE O MONITOR, CONSULTAR (7,8,9,10,11). UM PONTO DO MONITOR QUE PODE SER CONSIDERADO NEGATIVO E' A SUA FALTA DE MODULARIDADE, O QUE DIFICULTA SERIAMENTE A INTRODUÇÃO DE QUAISQUER MODIFICAÇÕES.

PARA TORNAR BEM CLARO O FUNCIONAMENTO DO MONITOR, VAMOS ACOMPANHAR A EXECUÇÃO DE UM PROGRAMA TÍPICO DE UM ALUNO DE PROGRAMAÇÃO FORTRAN. CADA PROGRAMA, QUE PARA O MONITOR E' UM "JOB", CONTEM ALÉM DO PROGRAMA FONTE EM FORTRAN E OS CARTÕES DE DADOS QUE PORVENTURA EXISTAM, CARTOES DE CONTROLE PARA O SISTEMA, QUE DETERMINAM QUAL A PRÓXIMA ETAPA A EXECUTAR (// JOB, // FOR, ETC...), ALÉM DE SERVIREM DE DELIMITADORES ENTRE O PROGRAMA FONTE E OS DADOS, E O PROGRAMA DE UM USUÁRIO E O DE OUTRO (FIG. 3).

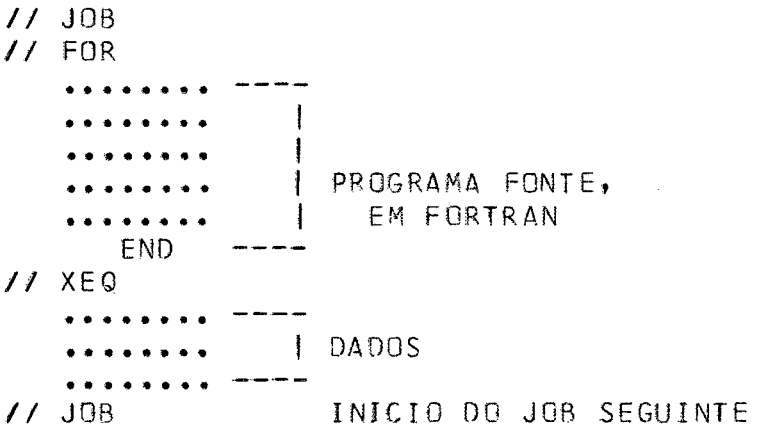


FIG. 3: UM JOB TÍPICO DO 1130

INICIALMENTE E' PROCESSADO O CARTAO DE CONTROLE "// JOB", CUJA FINALIDADE PRINCIPAL E' A DE SEPARAR UM JOB DE UM USUÁRIO DO DE OUTRO, IMPEDINDO QUE HAJA INTERAÇÃO ENTRE OS DOIS. NESTA ETAPA, QUE E' REALIZADA PELO SUPERVISOR, E' INICIALIZADA UMA SÉRIE DE TABELAS DE CONTROLE RESIDENTES EM DISCO. A SEGUIR, O SUPERVISOR INTERPRETA O CARTAO "// FOR", PASSANDO O CONTROLE PARA O COMPILADOR FORTRAN.

O COMPILADOR FORTRAN COMPILA O PROGRAMA FONTE, PONDO O PROGRAMA OBJETO NA AREA DE TRABALHO, EM DISCO. O CONTROLE E' DEVOLVIDO AO SUPERVISOR, QUE INTERPRETA O CARTAO "// XEQ", PASSANDO O CONTROLE PARA O "CORE LOAD BUILDER", VISTO QUE O PROGRAMA OBJETO AINDA NAO ESTA EM FORMA EXECUTÁVEL.

O "CORE LOAD BUILDER" PROCESSA O PROGRAMA OBJETO, TORNANDO-O EXECUTÁVEL, ATRAVÉS DA LIGAÇÃO COM OS SUBPROGRAMAS REFERENCIADOS. EM SEGUIDA, O CONTROLE E' TRANSFERIDO PARA O "CORE IMAGE LOADER", QUE CARREGA O PROGRAMA OBJETO NA MEMÓRIA E LHE TRANSFERE O CONTROLE. ESTE POR SUA VEZ LÊ OS CARTÕES DE DADOS, IMPRIME OS RESULTADOS E O CONTROLE E' DEVOLVIDO AO SUPERVISOR, PARA REINICIAR O CICLO COM O JOB SEGUINTE (FIG. 4).

AS ETAPAS DEMORADAS NO PROCESSO DESCRITO ACIMA SÃO: O PROCESSAMENTO DO JOB, A COMPILAÇÃO E O PROCESSAMENTO DO "CORE LOAD BUILDER". TODAS ESTAS ETAPAS ENVOLVEM GRANDE UTILIZAÇÃO DO DISCO, QUE E' RELATIVAMENTE LENTO NO 1130. A COMPILAÇÃO, EM PARTICULAR, E' COMPOSTA DE 28 FASES, ONDE CADA UMA TERÁ DE SER LIDA DO DISCO.

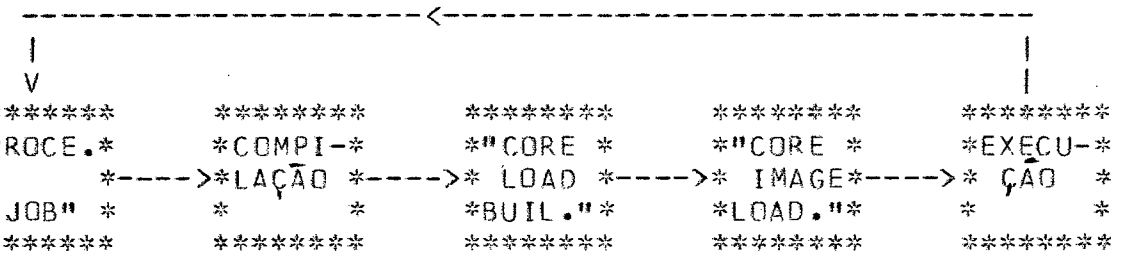


FIG. 4: FLUXO LÓGICO DO CONTROLE
DE UM PROGRAMA NO MONITOR

****
* *
* CAPITULO IV * ESTRUTURAÇÃO INICIAL *
* *
****

ANTES DE INICIARMOS A ELABORAÇÃO DO SISTEMA, IREMOS TECER ALGUMAS CONSIDERAÇÕES INICIAIS QUE IRÃO TENTAR DEFINIR MELHOR AS CARACTERÍSTICAS DO MESMO.

1. UTILIZAÇÃO DO DISCO

UM BOM COMEÇO PARA O DESENVOLVIMENTO DO SISTEMA E' ANALISAR O MONITOR E VERIFICAR QUAIS SÃO OS PONTOS DE POSSIVEIS OTIMIZAÇÕES, PARA O PROCESSAMENTO DE PROGRAMAS SIMPLES. A NOSSA HIPÓTESE E' DE QUE O PRINCIPAL DESTES PONTOS E' A REDUÇÃO NA UTILIZAÇÃO DO DISCO, MUITO UTILIZADO PELO MONITOR, TENDO EM VISTA QUE O DISCO DO 1130 E' RELATIVAMENTE LENTO. ESTA AFIRMAÇÃO E' EQUIVALENTE A DIZERMOS QUE O SISTEMA "COPPE-FORTRAN" DEVE SER ELABORADO PARA QUE DENTRO DAS PARTICULARIDADES DOS PROGRAMAS JÁ DESCRITOS, UTILIZE AO MÍNIMO O DISCO, E PROCESSE O CONJUNTO DESTES PROGRAMAS COM MAIOR EFICIÊNCIA.

2. CARTÕES DE CONTROLE

ASSIM COMO NO MONITOR, NO "COPPE-FORTRAN" TAMBÉM NECESSITAREMOS DE TER CARTÕES DE CONTROLE PARA SEPARAR O PROGRAMA DE UM USUÁRIO DO DE OUTRO, SEPARAR O PROGRAMA FONTE DOS DADOS, ETC.... PARA ESTES CARTÕES DE CONTROLE, PODEREMOS UTILIZAR OS MESMOS DO MONITOR, OU CRIAR OUTROS, DIFERENTES. CRIAR OUTROS NOVOS, ALEM DE NÃO TRAZER NENHUM BENEFÍCIO, TRARIA O GRANDE PROBLEMA DE QUE OS USUÁRIOS TERIAM DE APRENDER NOVOS CARTÕES DE CONTRÔLE, ALEM DE TORNAR OS DOIS SISTEMAS INCOMPATÍVEIS. DECIDIMOS POIS, UTILIZAR OS MESMOS CARTÕES DE CONTROLE DO MONITOR.

3. SEQÜÊNCIA DE OPERACOES

DESCREVEREMOS A SEGUIR, COMO SERIAM REALIZADAS AS VARIAS ETAPAS DO PROCESSAMENTO DE UM PROGRAMA, DESCRITO NO FINAL DO CAPITULO III, POR INTERMÉDIO DO "COPPE-FORTRAN".

AS TABELAS DE CONTROLE NECESSÁRIAS DURANTE O PROCESSAMENTO DO CARTÃO "JOB" SAO RESIDENTES NA MEMORIA, DE MODO QUE ESTA ETAPA SEJA REALIZADA RAPIDAMENTE.

O SISTEMA TAMBEM NECESSITARÁ DE UM COMPILADOR FORTRAN PARA A TRADUÇÃO DOS PROGRAMAS ESCRITOS EM FORTRAN PARA A LINGUAGEM DE MÁQUINA. NAO PODEMOS UTILIZAR O DO MONITOR, POIS ESTE NÃO CONTEM AS OTIMIZAÇÕES QUE PRETENDEMOS OBTER. TEMOS POIS DE ELABORAR UM COMPILADOR PRÓPRIO. A ALTERNATIVA DE UTILIZAR UM INTERPRETADOR NAO E' BOA, VISTO QUE E' BEM MAIS LENTO DO QUE UM COMPILADOR. SOBRE AS CARACTERÍSTICAS DO COMPILADOR, VOLTAREMOS A FALAR MAIS TARDE. PARA AUMENTAR A VELOCIDADE DO COMPILADOR, DEVEMOS COLOCÁ-LO (SE POSSÍVEL) INTEGRALMENTE NA MEMORIA, AFIM DE EVITAR A UTILIZAÇÃO DO DISCO PARA FAZER "OVERLAYS".

NESTE PONTO, PARA PODERMOS PROSSEGUIR COM AS OTIMIZAÇÕES DESEJADAS, TEREMOS DE ACEITAR ALGUMAS RESTRIÇÕES SOBRE OS RECURSOS QUE O SISTEMA OFERECERÁ. A PRIMEIRA RESTRIÇÃO TEM COMO OBJETIVO EVITAR O USO DO DISCO DURANTE O PROCESSAMENTO DO "CORE LOAD BUILDER". SE ACEITARMOS A RESTRIÇÃO DE PROIBIR O USO DE SUBPROGRAMAS GUARDADOS NA BIBLIOTECA DO SISTEMA AO USUARIO (O QUE E' RAZOÁVEL PARA PROGRAMAS SIMPLES), PODEMOS ADOTAR O SEGUINTE ESQUEMA: AS FUNÇÕES E SUBROTINAS PADRÕES DA LINGUAGEM ESTARIAM PRESENTES NA MEMORIA INTERNA, DURANTE A EXECUÇÃO COM ENDERECOS CONHECIDOS DURANTE A COMPILAÇÃO; O PROGRAMA OBJETO GERADO PELO COMPILADOR SERIA ABSOLUTO E GERADO NA PROPRIA MEMÓRIA INTERNA E AS REFERÊNCIAS ENTRE AS FUNÇÕES E SUBROTINAS DEFINIDAS PELO USUÁRIO SERIAM RESOLVIDAS PELO PRÓPRIO COMPILADOR. COM ESTE ESQUEMA SIMPLESMENTE NAO NECESSITARÍAMOS MAIS DO "CORE LOAD BUILDER" NEM DO "CORE IMAGE LOADER". NOTE-SE QUE PERDEMOS UM POUCO EM GENERALIDADE (NO CASO O ACESSO A PROGRAMAS DA BIBLIOTECA) PARA GANHARMOS EM VELOCIDADE. ESTE COMPROMISSO OCORRERÁ MAIS VEZES DURANTE ESTE TRABALHO.

FINDA A COMPILAÇÃO, O CONTRÔLE E' PASSADO DIRETAMENTE AO PROGRAMA OBJETO, APÓS A INTERPRETAÇÃO DO CARTAO "// XEQ". APOS A EXECUÇÃO DO PROGRAMA, SERÁ LIDO NOVO CARTAO "// JOB", E O CICLO SE REPETE (FIG. 5).

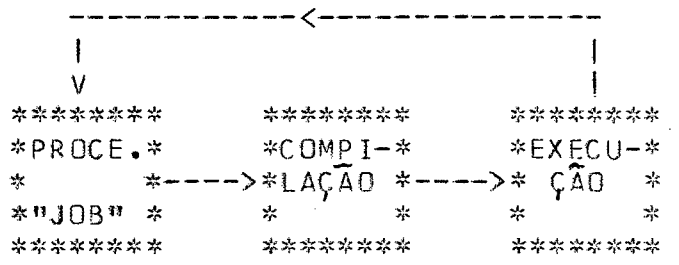


FIG. 5: FLUXO LÓGICO DO CONTROLE
DE UM PROGRAMA NO COPPE-FORTRAN

4. FASES DO SISTEMA

BASEADO NAS OPERAÇÕES DO SISTEMA QUE ACABAMOS DE DESCREVER, PODEMOS DIVIDIR O SISTEMA EM 3 PARTES, DE ACORDO COM A SUA FINALIDADE: A FASE DE SUPERVISÃO, A FASE DE COMPILAÇÃO E A FASE DE EXECUÇÃO.

A FASE DE SUPERVISÃO, OU SUPERVISOR CONSISTE DOS PROGRAMAS NECESSÁRIOS PARA REALIZAR A ANÁLISE E INTERPRETAÇÃO DOS CARTÕES DE CONTROLE, COMO O PROCESSAMENTO DO JOB, E A LIGAÇÃO ENTRE AS OUTRAS FASES.

A FASE DE COMPILAÇÃO, CONSISTE DO COMPILADOR PROPRIAMENTE DITO, QUE IRA GERAR UM PROGRAMA OBJETO A PARTIR DE CADA PROGRAMA FORTRAN.

A FASE DE EXECUÇÃO CONSISTE DE TODOS AS ROTINAS NECESSÁRIAS DURANTE A EXECUÇÃO DO PROGRAMA OBJETO. ENTRE ESTAS, TEMOS AS ROTINAS ARITMÉTICAS PARA OPERAÇÕES REAIS E INTEIRAS, AS FUNÇÕES E SUBROTINAS PADRÕES DA LINGUAGEM (SIN, ALOG, ETC..), E AS ROTINAS DE INTERPRETAÇÃO DO FORMAT.

5. UNIDADES DE ENTRADA/SAIDA

OUTRO PONTO A SER CONSIDERADO VERSA SOBRE AS UNIDADES DE ENTRADA/SAIDA QUE DEVEM SER PERMITIDAS DURANTE A EXECUÇÃO DE UM PROGRAMA. OS COMPROMISSOS ENVOLVIDOS NESTA CONSIDERAÇÃO SAO OS SEGUINTES: SE SUPORTARMOS MUITAS UNIDADES DE ENTRADA/SAIDA, OCUPAREMOS MUITO A MEMÓRIA INTERNA COM AS SUBROTINAS NECESSÁRIAS PARA AS UNIDADES. ALEM DISTO, MUITAS DESTAS UNIDADES DE ENTRADA/SAIDA SAO BEM LENTAS, COMO O PLOTTER, E SE SUPORTADAS IRAO TORNAR O SISTEMA MAIS LENTO. POR OUTRO LADO, QUANTO MENOS UNIDADES SUPORTARMOS, MENOS GERAL RESULTARÁ O SISTEMA. COMO CRITÉRIO DE DECISÃO, RESOLVEMOS SACRIFICAR NOVAMENTE A GENERALIDADE EM FAVOR DA VELOCIDADE, NOTANDO QUE A QUASE TOTALIDADE DOS PROGRAMAS DE PRINCIPIANTES UTILIZAM APENAS UNIDADES SIMPLES. DECIDIMOS POIS QUE O SISTEMA SUPORTARÁ APENAS UMA LEITORA DE CARTOES (2501 OU 1442) E UMA IMPRESSORA (1403 OU 1132).

6. MÉTODO DE IMPLEMENTAÇÃO

PARA A IMPLEMENTAÇÃO DO SISTEMA, ACHAMOS CONVENIENTE ELABORAR A ESTRUTURA E EXECUTAR A PROGRAMAÇÃO EM PARALELO. SOMOS DE OPINIÃO QUE E' MUITO DIFICIL ELABORAR A ESTRUTURA COMPLETA E SÓ DEPOIS INICIAR A PROGRAMAÇÃO, EM VIRTUDE DE QUE A ESTRUTURA E' MUITO DEPENDENTE DE DETALHES DE PROGRAMAÇÃO, QUE NÃO PODEM SER INTEIRAMENTE PREVISTOS. TRATA-SE MUITO MAIS DE UM PROCESSO ITERATIVO EM QUE A CADA PASSO DA PROGRAMACAO DESCOBRE-SE NOVAS IDÉIAS PARA A ESTRUTURA, OU NELA DESCOBREM-SE DEFEITOS.

7. LINGUAGEM USADA PARA A IMPLEMENTAÇÃO

SOBRE A LINGUAGEM QUE IRÁ SER UTILIZADA PARA A IMPLEMENTAÇÃO DO SISTEMA SÓ HÁ DUAS OPÇÕES: FORTRAN OU ASSEMBLER. ENTRE ESTES DOIS NAO E' DIFICIL DE APONTAR O ASSEMBLER, POIS EMBORA A PROGRAMAÇÃO NESTA LINGUAGEM SEJA MAIS TRABALHOSA, A LINGUAGEM FORTRAN DO 1130 E' RESTRITA DEMAIS PARA A TAREFA, ALEM DE QUE FORNECE UM PROGRAMA OBJETO MUITO INEFICIENTE (MAIS TARDE SURGIRAM O RPG E O COBOL, QUE TAMBEM SAO INFERIORES AO ASSEMBLER PARA A FUNÇÃO).

8. LINGUAGEM FORTRAN ACEITA PELO SISTEMA

E' INTERESSANTE QUE A LINGUAGEM FORTRAN ACEITA PELO SISTEMA SEJA COMPATÍVEL COM O MONITOR, PARA QUE HAJA A COMPATIBILIDADE ENTRE OS SISTEMA. APESAR DISTO, COMO O SISTEMA E' ORIENTADO PARA PRINCIPIANTES, NADA CUSTA FAZER CERTAS EXTENSÕES A LINGUAGEM, COMO COMANDOS DE ENTRADA/SAIDA SEM FORMATO, DE VALOR INESTIMÁVEL PARA O ENSINO. INFELIZMENTE, EM DETRIMENTO DA COMPATIBILIDADE, EM VIRTUDE DAS RESTRIÇÕES ANTERIORMENTE MENCIONADAS SOBRE AS UNIDADES DE ENTRADA/SAIDA, ALGUNS COMANDOS NAO PODEM SER ACEITOS. SAO ELES: "DEFINE FILE", "REWIND", "END FILE", "BACKSPACE", "FIND" E COMANDOS "READ" E "WRITE" PARA O DISCO.

9. CÓDIGOS DE CARACTERES UTILIZADOS

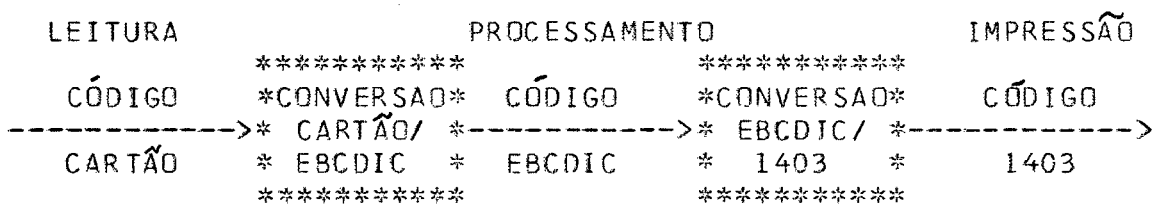
EM UM PROGRAMA TÍPICO DE INFORMAÇÃC NAO NUMÉRICA, ISTO E', QUE OPERA COM CARACTERES E SIMBOLOS COMO E' O CASO DO COPPE-FORTRAN, AS INFORMAÇÕES TEM DE SER REPRESENTADAS NO INTERIOR DA MEMORIA ATRAVES DE UM CODIGO DE CARACTERES. ISTO SGINIFICA ASSOCIAR A CADA CARACTER UTILIZADO UMA CONFIGURACAO DE BITS TAL QUE ELE POSSA SER REPRESENTADO INTERNAMENTE NO COMPUTADOR.

NO COMPUTADOR IBM-1130 JÁ SÃO UTILIZADOS VÁRIOS CÓDIGOS DE CARACTERES, POIS QUASE TODAS AS UNIDADES DE ENTRADA/SAIDA UTILIZAM UMA REPRESENTAÇÃO DIFERENTE PARA OS CARACTERES TRANSMITIDOS. CONSIDERANDO APENAS AS UNIDADES QUE IREMOS SUPORTAR, TEMOS OS SEGUINTES CODIGOS: O CODIGO DE CARTÃO (PROVENIENTE DAS LEITORAS DE CARTÃO 2501 E 1442 NA ÁREA DE LEITURA DE UM CARTÃO), O EBCDIC (UTILIZADO PELA 1132) E O CÓDIGO DA 1403 (ESTES DOIS UTILIZADOS NA ÁREA DE IMPRESSÃO DE UMA LINHA). COMO TEMOS CODIGOS DIFERENTES NAS UNIDADES DE ENTRADA/SAIDA, E CERTAMENTE IREMOS DESEJAR IMPRIMIR ALGUNS DOS CARTÕES LIDOS, SERA NECESSÁRIA REALIZAR UMA CONVERSÃO ENTRE OS CÓDIGOS DE ENTRADA E SAÍDA.

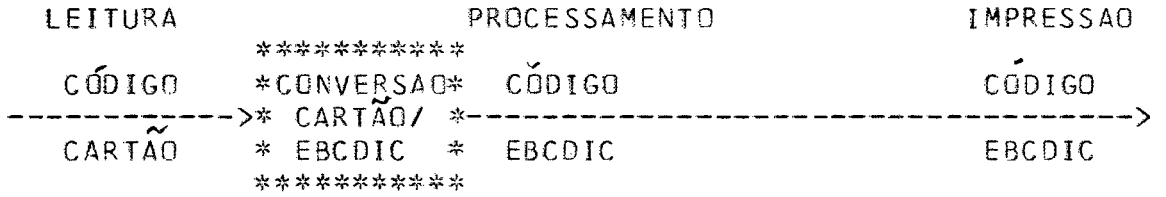
PARA A UTILIZAÇÃO NO SISTEMA, TEMOS DE ESCOLHER TAMBÉM UM CÓDIGO DE CARACTERES. E' INTERESSANTE UNIFORMIZAR ESTE CÓDIGO EM TODO SISTEMA, PARA QUE NAO HAJA CONVERSÕES DESNECESSÁRIAS. NATURALMENTE, A NÃO SER QUE OS TRÊS ACIMA MENCIONADOS SE MOSTREM MUITO INADEQUADOS, NAO E' DESEJAVEL INTRODUZIR UM QUARTO CÓDIGO, QUE TAMBÉM INTRODUZIRIA CONVERSOES DESNECESSÁRIAS (NO CASO ENTRADA PARA INTERNO E INTERNO PARA IMPRESSAO).

DE INÍCIO E' VÁLIDO ABANDONAR O CODIGO DE CARTÃO PARA NAO OCUPAR MEMÓRIA DESNECESSARIA, POIS ELE POSSUI 12 BITS POR CARACTER, COMPARADO COM OS 8 DO EBCDIC E OS 7 DA 1403. QUALQUER QUE ESCOLHAMOS DESTES DOIS ULTIMOS, TEREMOS DE FAZER UMA CONVERSÃO DUPLA - SE UTILIZARMOS O EBCDIC TEREMOS DE FAZER UMA CONVERSÃO DUPLA AO UTILIZAR A IMPRESSORA 1403, E SE UTILIZARMOS O CODIGO DA 1403, TEREMOS DE FAZER UMA CONVERSÃO DUPLA AO UTILIZAR A IMPRESSORA 1132.

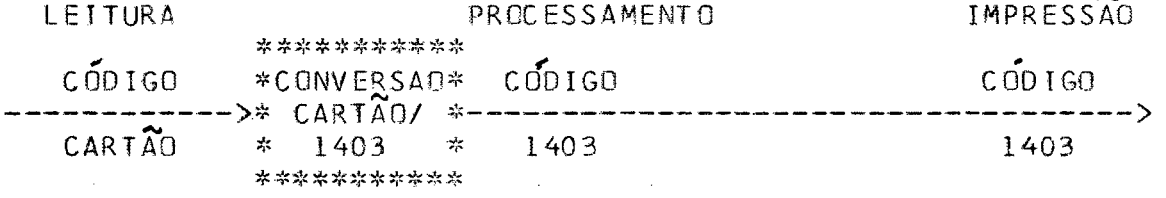
DOS DOIS CÓDIGOS ESCOLHEMOS O CODIGO DA 1403 AO INVÉS DO EBCDIC, POIS A IMPRESSORA 1403 E' BEM MAIS RÁPIDA, LOGO MAIS SUJEITA A ATRASOS EM VIRTUDE DE CONVERSÕES (FIG. 6).



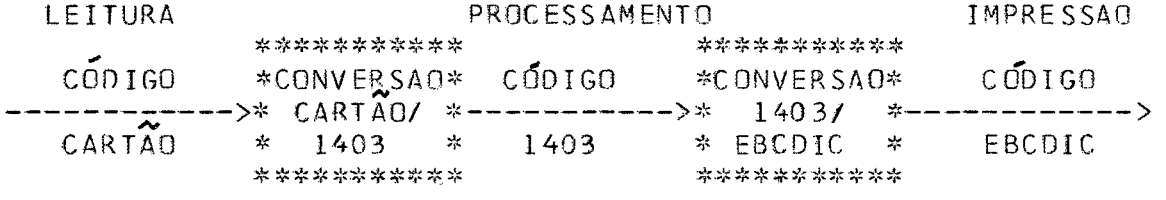
A. PELO MONITOR, COM A 1403



B. PELO MONITOR, COM A 1132



C. PELO COPPE-FORTRAN, COM A 1403



D. PELO COPPE-FORTRAN, COM A 1132

FIG. 6: CONVERSÕES DE CÓDIGOS

10. ANÁLISE LÉXICA

PARA A ANÁLISE LÉXICA, PARTE INDISPENSÁVEL EM QUALQUER COMPILADOR, NECESSITAMOS DE UMA ROTINA QUE EXTRAIA OS CARCTERES DO PROGRAMA FONTE. A ANÁLISE LÉXICA PODE SER ENORMEMENTE FACILITADA SE ESTA ROTINA FORNECER ASSOCIADO A CADA CARACTER, O SEU TIPO, ISTO E', INFORMANDO SE O REFERIDO CARACTER E' UM ALGARISMO, LETRA OU CARACTER ESPECIAL. ESTA ROTINA PODE INCLUSIVE PULAR AUTOMATICAMENTE AS COLUNAS EM BRANCO, QUE SAO TRANSPARENTES EM FORTRAN, E OCUPAR-SE COM OS CARTÕES DE CONTINUAÇÃO. ESTA ROTINA FOI INTITULADA "GETCH", E SERÁ UMA DAS PRIMEIRAS A SER FEITA.

OUTRAS ROTINAS IMPORTANTES A DESTACAR SAO: "NARET", PARA RETIRAR IDENTIFICADORES, "DECBN" PARA RETIRAR CONSTANTES INTEIRAS E TRANSFORMÁ-LAS PARA BINÁRIO. E' INTERESSANTE MENCIONAR QUE TODAS AS ROTINAS DE ANÁLISE LÉXICA LÊEM O CARACTER INICIAL DO ELEMENTO SEGUINTE, E O GUARDAM NA AREA DE COMUNICAÇÕES. DESTE MODO, AO INICIAR A ANÁLISE DE UM NOVO ELEMENTO, O SEU PRIMEIRO CARACTER PODE SER RETIRADO DA AREA DE COMUNICAÇÕES. ÊSTE ESQUEMA E' UTILIZADO, POIS NA RETIRADA DA MAIORIA DOS ELEMENTOS, O SEU FINAL SÓ E' DETETADO QUANDO E' LIDO O PRIMEIRO CARACTER DO ELEMENTO SEGUINTE.

11. ANÁLISE SINTAICA

VOLTAREMOS AGORA PARA A DISCUSSÃO SOBRE O TIPO DE COMPILADOR A UTILIZAR. TEMOS BASICAMENTE DOIS TIPOS: O DE SINTAXE IMPLÍCITA, E O DIRIGIDO POR TABELAS DE SINTAXE. AS CARACTERISTICAS EM RESUMO SAO: O DE SINTAXE IMPLICITA E' MAIS RÁPIDO, MAS E' MAIS DIFÍCIL DE ALTERAR A SINTAXE DA LINGUAGEM RECONHECIDA. O DIRIGIDO POR TABELAS TEM CARACTERÍSTICAS INVERSAS: ENQUANTO E' MAIS FLEXÍVEL, PERMITINDO FACILMENTE MODIFICAÇÕES NA SINTAXE DA LINGUAGEM, PERDE MUITO NA VELOCIDADE DE COMPILAÇÃO. PARA O NOSSO CASO, A ESCOLHA NAO E' DIFÍCIL: COMO O NOSSO OBJETIVO PRINCIPAL E' A VELOCIDADE, E A NOSSA LINGUAGEM E' FIXA (O FORTRAN), O COMPILADOR DE SINTAXE IMPLÍCITA E' CERTAMENTE MAIS CONVENIENTE.

PELOS MESMOS MOTIVOS, O COMPILADOR DEVERA (SE POSSÍVEL) SER DE APENAS UM PASSO, ISTO E', "EXAMINAR" O PROGRAMA FONTE UMA ÚNICA VEZ. COMO A LINGUAGEM FORTRAN PERMITE ISTO, E UM COMPILADOR MULTI-PASSO E' MAIS LENTO, ESTA DECISÃO PARECE SER BEM FUNDAMENTADA.

EXAMINANDO A LINGUAGEM FORTRAN, PODEMOS VER QUE RETIRANDO AS EXPRESSOES ARITMETICAS, O RESTANTE PODE SER COMPILADO POR TÉCNICAS DE MAQUINAS DE ESTADO FINITO.

12. DEPURAÇÃO

COM EXPERIÊNCIAS ANTERIORES, FOI VERIFICADO QUE EM PROGRAMAS COMPLEXOS ESCRITOS EM ASSEMBLER (COMO TAMBEM E' O CASO PRESENTE) E' MUITO IMPORTANTE PROVER EFICIENTES MEIOS DE DEPURAÇÃO, PARA EVITAR QUE A PESQUISA DA CAUSA DE UM ERRO DEMORE UM TEMPO MUITO LONGO. UM RECURSO QUE CERTAMENTE AUXILIA MUITO A DEPURAÇÃO DE UM PROGRAMA EM LINGUAGEM SIMBÓLICA E' UM PROGRAMA DE "TRACE", QUE EMITE RELATORIOS INDICANDO CADA INSTRUÇÃO DE MÁQUINA EXECUTADA, JUNTAMENTE COM OS RESPECTIVOS CONTEÚDOS DOS DIVERSOS REGISTROS, DURANTE A EXECUÇÃO DE UM DETERMINADO TRECHO DESTE PROGRAMA EM LINGUAGEM SIMBÓLICA. DECIDIMOS PORTANTO ELABORAR UM PROGRAMA DE "TRACE" COM ESTAS CARACTERISTICAS.

DURANTE A IMPLEMENTAÇÃO DO SISTEMA, CERTAMENTE EM CERTO PONTO SURGIRIA A NECESSIDADE DE ESCOLHER O ALGORITMO MAIS RÁPIDO A UTILIZAR PARA UMA DETERMINADA FUNÇÃO, ENTRE UM CERTO NÚMERO DE CANDIDATOS. NEM SEMPRE UM CÁLCULO TEÓRICO E' POSSÍVEL OU SIMPLES, E UMA SOLUÇÃO PARA ISTO E' A MEDIÇÃO NA PRÁTICA DO TEMPO GASTO PELOS VARIOS ALGORITMOS PROPÓSTOS. COMO O 1130 NAO TEM RELÓGIO INTERNO, DECIDIMOS ELABORAR UMA ROTINA DE MEDIÇÃO DE TEMPO BASEADA NO TEMPO MEDIO DE CADA INSTRUÇÃO DE MÁQUINA, UTILIZANDO O RECURSO DO 1130 DE PODER, SE DESEJADO, GERAR UMA INTERRUPÇÃO NO FINAL DE CADA INSTRUÇÃO DE MÁQUINA EXECUTADA.

PASSAREMOS AGORA, A DESCREVER MAIS DOIS RECURSOS QUE FORAM JULGADOS NECESSÁRIOS PARA AUXILIAR A DEPURAÇÃO DO SISTEMA. O PRIMEIRO DESTES E' UMA SUBROTINA CUJA FUNÇÃO E' A DE IMPRIMIR O CONTEÚDO DA TABELA DE SIMBOLOS, EM UMA FORMA FÁCIL DE SER CONSULTADA, CONTENDO OS SIMBOLOS UTILIZADOS PELO PROGRAMA FONTE E AS RESPECTIVAS ATRIBUIÇÕES FEITAS PELO COMPILADOR.

O SEGUNDO RECURSO E' A IMPLEMENTAÇÃO DE 2 COMANDOS, "DUMP COMP" E "DUMP EXEC", NATURALMENTE NAO CONSTANTES NA LINGUAGEM FORTRAN, MAS QUE SERIAM ACEITOS PELO COMPILADOR. A FUNÇÃO DESTES COMANDOS SÃO RESPECTIVAMENTE, A IMPRESSÃO DE AREAS DA MEMORIA, EM FORMATO HEXADECIMAL, DURANTE A COMPILAÇÃO E A EXECUÇÃO DE UM PROGRAMA. COM ESTAS LISTAGENS, PODE-SE COMPARAR, POR EXEMPLO UMA DETERMINADA REGIÃO DA MEMORIA ANTES E DEPOIS DA COMPILAÇAO DE UM DETERMINADO COMANDO, E PORTANTO ACOMPANHAR A COMPILAÇÃO.

13. RELAÇÃO ENIRE O SISIEMA E O MONITOR

O SISTEMA, DO PONTO DE VISTA DO MONITOR, E' APENAS UM PROGRAMA NORMAL, EXECUTADO ATRAVÉS DE UM CARTÃO "// XEQ". AO INICIAR A EXECUÇAO DO SISTEMA, ELE ASSUME O CONTROLE, EXECUTANDO TODOS OS JOB'S ENCONTRADOS, ATÉ QUE SEJA LIDO UM DELIMITADOR ESPECIAL, QUANDO ENTÃO O CONTROLE E' DEVOLVIDO AO MONITOR. ADOTAMOS ESTA SOLUÇÃO, POIS ACHAMOS QUE A CRIAÇÃO DE UM CARTÃO ESPECIAL (POR EXEMPLO "// CFOR") ENVOLVERIA GRANDES ALTERAÇÕES NO MONITOR, SEM OBTER NENHUMA VANTAGEM SUBSTANCIAL.

CAPITULO V * TÉCNICAS DE COMPILAÇÃO*

1. MONTAGEM DO SISTEMA

PARA A ELABORAÇÃO DO SISTEMA, E' MUITO INTERESSANTE TORNÁ-LO BEM MODULAR, ISTO E', SUBDIVIDI-LO EM MUITAS PARTES INDEPENDENTES, COM FUNÇÕES PRÓPRIAS E EXCLUSIVAS, PARA QUE QUANDO FOR DESEJADO REALIZAR QUALQUER CONSÊRTO OU MODIFICAÇÃO, APENAS SEJA NECESSÁRIO REFERIR-SE A PARTE QUE TRATA DA FUNÇÃO EM QUESTÃO, AO INVÉS DE TER DE ALTERAR DIVERSOS PONTOS DO SISTEMA, CASO SE CADA UMA DAS FUNÇÕES FOSSE DILUÍDA POR TODO O SISTEMA. OS COMPROMISSOS ENVOLVIDOS PELA MODULARIDADE SÃO: A PERDA DE UM POUCO DA VELOCIDADE E UM SUBSTANCIAL AUMENTO NA FACILIDADE DE PROGRAMAÇÃO. EM TERMOS DE PROGRAMAÇÃO, A MODULARIDADE PODE SER CONSEGUIDA ATRAVÉS DO INTENSO USO DE SUBROTINAS, ONDE CADA SUBROTINA TERIA UMA FINALIDADE BEM DETERMINADA.

A. MONTAGEM INDIVIDUAL

PARA CONCRETIZAR ESTA IDÉIA, O MAIS NATURAL SERIA MONTAR-SE CADA SUBROTINA ISOLADAMENTE, COLOCANDO CADA PROGRAMA OBJETO RELOCÁVEL RESULTANTE NA BIBLIOTECA DE PROGRAMAS, E EM SEGUIDA COM O AUXÍLIO DO "CORE LOAD BUILDER", FORMAR O MODULO DE CARGA, QUE SERIA UMA DAS FASES DO SISTEMA (FIG. 7).

ESTE MODO, QUE E' O DE MAIS FÁCIL MANUTENÇÃO, TEM GRAVES DEFEITOS QUE PRÀTICAMENTE IMPEDEM O SEU USO, DEVIDO AS RESTRIÇÕES DO "CORE LOAD BUILDER". O "CORE LOAD BUIDER" PERMITE APENAS REFERENCIAS EXTERNAS DO TIPO DE DESVIO (CALL), QUE SAO INSUFICIENTES, POIS NECESSITAMOS REFERÊNCIAS EM QUE OPERAMOS COM ARGUMENTOS EXTERNOS, ALEM DE POSSUIR RECURSOS MUITO LIMITADOS PARA "OVERLAYS", EM QUE NAO SE PODE

ESCOLHER A ORDEM DE COLOCAÇÃO DAS ROTINAS, NEM UTILIZAR MAIS DE UM NÓ.

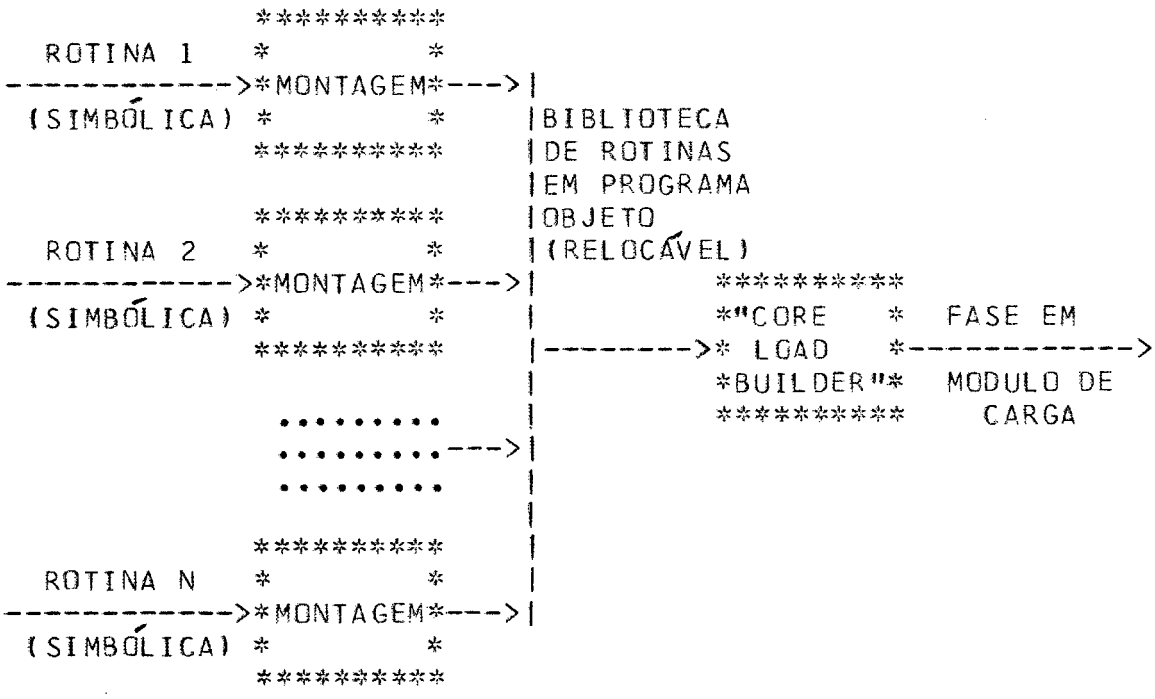


FIG. 7: MONTANDO CADA ROTINA SEPARADAMENTE
(COM OS RECURSOS DO "CORE LOAD BUILDER")

8. MONTAGEM CONJUNTA

UMA OUTRA SOLUÇÃO, E' NAO UTILIZAR OS RECURSOS DO "CORE LOAD BUILDER", MONTANDO CADA FASE INTEGRALMENTE, COM ORIGEM ABSOLUTA, E RESOLVENDO OS SIMBOLOS EXTERNOS AS ROTINAS "MANUALMENTE", POR MEIO DE CARTÕES "EQU", E PROGRAMANDO OS "OVERLAYS" DURANTE A PRÓPRIA MONTAGEM COM O AUXÍLIO DE UMA ROTINA ESPECIAL. ESTA SOLUÇÃO, EMBORA SUPRA PERFEITAMENTE AS NECESSIDADES E GANHA MUITO EM FLEXIBILIDADE, PERDE EM FACILIDADE DE MODIFICAÇÃO. ALEM DISSO AUMENTA CONSIDERAVELMENTE OS TEMPOS DE MONTAGEM, VISTO QUE PARA QUALQUER MODIFICAÇÃO SERÁ NECESSÁRIO MONTAR UMA FASE INTEIRA NOVAMENTE (FIG. 8).

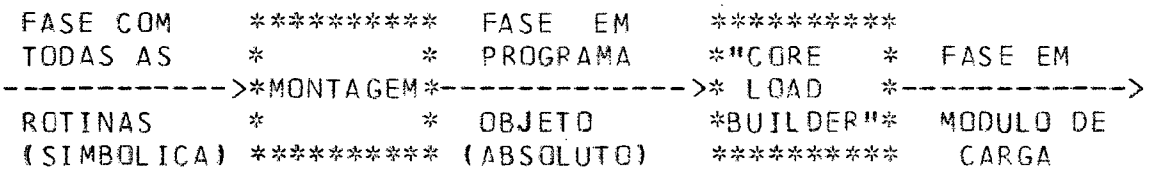


FIG. 8: MONTANDO A FASE COM TODAS SUAS ROTINAS INTEGRALMENTE (SEM OS RECURSOS DO "CORE LOAD BUILDER")

ENTRE OS DOIS MODOS APRESENTADOS, FOI ESCOLHIDO O SEGUNDO, POIS AS RESTRIÇÕES IMPOSTAS PELO PRIMEIRO O INVALIDAM. ESTE SEGUNDO MODO FOI UTILIZADO APENAS POR ALGUM TEMPO, POIS MAIS TARDE FOI LIBERADO O MACRO-ASSEMBLER, QUE PERMITE UMA SOLUÇÃO MELHOR DO QUE AS DUAS APRESENTADAS, E QUE SERA VISTA NO CAPITULO 6.

2. ÁREA DE COMUNICAÇÕES

PARA FACILITAR A COMUNICAÇÃO E PASSAGEM DE ARGUMENTOS ENTRE AS MUITAS SUBROTINAS QUE IRÃO FORMAR O SISTEMA, DECIDIU-SE PROVER UMA AREA DE COMUNICAÇÕES EM QUE SERIAM POSTAS AS TABELAS NECESSÁRIAS POR TODO O SISTEMA. NESTA AREA SERIAM COLOCADOS POR EXEMPLO, AS TABELAS NECESSÁRIAS DURANTE O PROCESSAMENTO DO JOB, E A TABELA DE CONVERSÃO DE CARACTERES. NA ÁREA SERIAM COLOCADOS TAMBÉM, TODAS AS PALAVRAS QUE REFLETEM O ESTADO DO SISTEMA NO MOMENTO, COMO POR EXEMPLO, OS PONTEIROS PARA O PROGRAMA OBJETO, PONTEIROS PARA TABELAS, ETC...

A ÁREA DE COMUNICAÇÕES CONTEM INFORMAÇÕES SOBRE OS SEGUINTES ITENS: CARTÕES DE CONTROLE DO COMPILADOR, TABELAS DO SISTEMA, ERROS COMETIDOS, PROGRAMA SENDO COMPILADO, ALOCAÇÃO DE MEMORIA, ENTRADA/SAIDA DE DADOS E CONTABILIDADE DO SISTEMA.

3. TABELAS

DURANTE A COMPILAÇÃO, E' NECESSÁRIA UMA SÉRIE DE TABELAS, PARA GUARDAR INFORMAÇÕES SOBRE O PROGRAMA EM COMPILAÇÃO, COMO OS IDENTIFICADORES UTILIZADOS, SUBPROGRAMAS REFERENCIADOS, ETC...

UMA DAS PRINCIPIAIS TABELAS E' A TABELA DE SIMBOLOS, QUE CONTEM OS IDENTIFICADORES UTILIZADOS PELO PROGRAMA, JUNTAMENTE COM SUAS RESPECTIVAS ATRIBUIÇÕES. ESTA TABELA E' MUITO CONSULTADA E ALTERADA DURANTE A COMPILAÇÃO, POIS QUALQUER VARIÁVEL, CONSTANTE, OU NÚMERO DE COMANDO ENCONTRADO NOS COMANDOS FORTRAN TEM DE SER NELA PESQUISADA, E EVENTUALMENTE INSERIDA. E' POIS IMPORTANTE MINIMIZAR AO MÁXIMO O TEMPO DE PROCURA NESTA TABELA. PARA EXECUTAR ESTA TAREFA, FORAM CONSIDERADAS AS TECNICAS LINEARES, BINÁRIAS E "HASH", E ACHAMOS QUE O PROCESSO MAIS INTERESSANTE E' A TECNICA "HASH", QUE E' RELATIVAMENTE SIMPLES DE IMPLEMENTAR, E E' BEM RÁPIDA. DOS VÁRIOS TIPOS DE "HASH", DECIDIMOS UTILIZAR O PROCESSO DA DIVISÃO, POR PROPICIAR DISTRIBUIÇÃO MAIS UNIFORME. NESTE PROCESSO, DIVIDE-SE O NÚMERO QUE REPRESENTA A VARIÁVEL NA CODIFICAÇÃO INTERNA DO COMPUTADOR PELO TAMANHO DA TABELA (QUE DEVE SER UM NUMERO PRIMO), E TOMA-SE O RESTO COMO ÍNDICE PARA A TABELA (13,14,15).

ALEM DA CITADA TABELA, TEMOS AINDA A TABELA DE OPERADORES E OPERANDOS ("STACK" ARITMETICO), EMPREGADO DURANTE A COMPILAÇÃO DAS EXPRESSÕES ARITMETICAS; A TABELA DE ARGUMENTOS, PARA CONTER OS ARGUMENTOS DE UM SUBPROGRAMA ORA SENDO COMPILADO; A TABELA DE PARAMETROS DO COMANDO "DO", PARA A COMPILAÇAO DOS COMANDOS "DO"; A TABELA DE VARIÁVEIS TEMPORÁRIAS, QUE SÃO NECESSÁRIAS DURANTE A GERAÇÃO DO PROGRAMA OBJETO REFERENTE AS EXPRESSÕES ARITMETICAS; A TABELA DE PROGRAMAS/SUBPROGRAMAS, PARA CONTER OS PROGRAMAS/SUBPROGRAMAS DEFINIDOS OU REFERENCIADOS E A TABELA DE PARAMETROS, USADA DURANTE A COMPILAÇÃO DE CHAMADAS A SUBPROGRAMAS.

4. IDENTIFICAÇÃO DOS COMANDOS

A IDENTIFICAÇÃO DOS COMANDOS FORTRAN, NECESSÁRIA DURANTE A COMPILACAO E' RELATIVAMENTE SIMPLES DE SER FEITA, COM AUXÍLIO DAS PALAVRAS-CHAVES, A MENOS DO COMANDO ARITMÉTICO. O COMANDO ARITMÉTICO, EM CERTOS CASOS PARTICULARES PODE SER CONFUNDIDO COM OUTROS COMANDOS, COMO NOS CASOS SEGUINTES:

```
DO 13 I = 50
READ (8,30) = 5.0
DATA = 3.0
```

O CRITÉRIO QUE DEVERA SER UTILIZADO PARA IDENTIFICAR O COMANDO E' O SEGUINTE:

A. EM PRIMEIRO LUGAR, VERIFICA-SE SE O COMANDO E' UM "FORMAT". SE OS 6 PRIMEIROS CARACTERES DE UM COMANDO FOREM 'FORMAT', ELE NAO PODERA DEIXAR DE SER UM "FORMAT", POIS O FORTRAN DO 1130 NAO PERMITE VARIAVEIS DE 6 CARACTERES.

B. EM SEGUIDA, VERIFICA-SE SE E' UM "DO". ESTE COMANDO E' IDENTIFICADO PELAS DUAS LETRAS INICIAIS, 'DO', E UMA VÍRGULA SEGUINDO UM SINAL DE IGUAL, SEM ABRE-PARENTESES ENTRE ELES.

C. NESTE PONTO JA PODEMOS IDENTIFICAR O COMANDO ARITMÉTICO, COMO POSSUINDO UM SINAL DE IGUAL FORA DE PARENTESES, E NENHUMA VÍRGULA FORA DE PARENTESES.

D. OS DEMAIS COMANDOS SÃO AGORA PESQUISADOS ATRAVÉS DE UMA TABELA CONTENDO AS PALAVRAS CHAVES.

PODEMOS VER QUE O ALGORÍTMO DESCRITO IDENTIFICA CORRETAMENTE OS COMANDOS ARITMÉTICOS DADOS ACIMA.

5. COMANDO EQUIVALENCE

UM DOS COMANDOS CUJA COMPILAÇÃO E' DAS MAIS COMPLEXAS, E' O "EQUIVALENCE", POIS ELE CONTEM MUITAS INTERRELAÇÕES EXPLÍCITAS E IMPLÍCITAS ENTRE AS VARIÁVEIS NELE REFERENCIADAS. POR EXEMPLO, NO COMANDO

```
EQUIVALENCE (A,B),(B,C)
```

TEMOS A RELAÇÃO EXPLICITA ENTRE "A" E "B", E ENTRE "B" E "C", E A RELAÇÃO IMPLICITA ENTRE "A" E "C", GERADA PELAS 2 RELAÇÕES EXPLICITAS ANTERIORES.

ENTRE OS VÁRIOS ALGORITMOS EXISTENTES PARA A COMPILAÇÃO DESTE COMANDO, COMO POR EXEMPLO O DE LISTAS CIRCULARES (13), E O DE ÁRVORES (16), DECIDIU-SE UTILIZAR ESTE ÚLTIMO, POR SER UM DOS MAIS RÁPIDOS, E PERMITIR FÁCIL DIAGNOSE DOS DIVERSOS ERROS. A IDÉIA BÁSICA DO ALGORITMO E' A CONSTRUÇÃO DE UMA ÁRVORE PARA CADA CONJUNTO DE VARIÁVEIS ENTRE PARENTESIS, EM QUE OS NÓS REPRESENTAM AS VARIÁVEIS (FIG. 9). ESTE ALGORITMO E' RELATIVAMENTE EXTENSO, SENDO POR ESTE MOTIVO OMITIDO.

EXEMPLO: PARA O COMANDO

EQUIVALENCE (A,B,C)

SERIA CONSTRUÍDA A ÁRVORE:

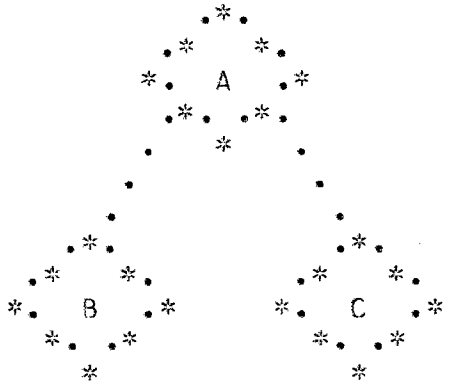


FIG. 9: ÁRVORE EMPREGADA EM EQUIVALENCE

6. COMPILAÇÃO DAS EXPRESSÕES ARITMÉTICAS

PARA A COMPILAÇÃO DAS EXPRESSOES ARITMETICAS, UM DOS PONTOS MAIS INTERESSANTES DO SISTEMA, FOI ADOTADO UM ALGORITMO BASEADO EM PRECEDÊNCIA DE OPERADORES, QUE DENTRE A VASTA LITERATURA, E' DOS MAIS CONVENIENTES, SIMPLES E RÁPIDO. TEM COMO DESVANTAGEM DIFICULTAR OTIMIZAÇÕES DO PROGRAMA OBJETO, O QUE NO NOSSO CASO NAO E' IMPORTANTE.

A IDÉIA DO ALGORITMO CONSISTE EM ASSOCIAR A CADA OPERADOR (+,-,*,/,**) UMA PRIORIDADE ESTÁTICA, QUE E' UM NUMERO INTEIRO E POSITIVO (NO NOSSO CASO 1,1,2,2,3). DEFINIMOS A SEGUIR UMA PRIORIDADE DINAMICA, COMO A SOMA DA PRIORIDADE ESTÁTICA E UM NÚMERO QUE INICIALMENTE VALE ZERO, E E' INCREMENTADO DE UM VALOR CONVENIENTE (NO NOSSO CASO 5), PARA CADA ABRE-PARENTESES É DECREMENTADO DO MESMO VALOR PARA CADA FECHA-PARENTESES ENCONTRADOS NA EXPRESSAO ARITMÉTICA. OS OPERADORES, JUNTAMENTE COM SUAS PRIORIDADES E OPERANDOS SAO COLOCADOS EM UM "STACK", ENQUANTO A PRECEDENCIA DINÂMICA DOS OPERADORES FOR CRESCENTE. QUANDO ENCONTRAMOS UM OPERADOR COM PRIORIDADE DINÂMICA IGUAL OU MENOR DO QUE O OPERADOR DO TOPO DO "STACK", A OPERAÇÃO DO TOPO DO "STACK" DEVE SER REALIZADA. E' GERADO O PROGRAMA OBJETO REFERENTE A ESTA OPERAÇÃO, E ELA E' RETIRADA DO "STACK". TENTA-SE NOVAMENTE INSERIR A OPERAÇÃO QUE ORIGINOU A GERAÇÃO DO CODIGO NO "STACK". ESTE PROCESSO VAI SENDO REPETIDO ATE' QUE SEJA ATINGIDO O FINAL DA EXPRESSÃO, E O "STACK" ESTEJA VAZIO (13,17).

7. DETEÇÃO DE DESVIOS INVALIDOS

COMO O SISTEMA DEVE FORNECER DIAGNOSTICOS BEM DETALHADOS, ACHAMOS INTERESSANTE INCLUIR UM ALGORITMO PARA A DETEÇÃO DE DESVIOS INVALIDOS PARA O INTERIOR DAS MALHAS DOS COMANDOS "DO", QUE FOI UTILIZADO PELO WATFOR. ESTE TIPO DE DETEÇÃO DE ERROS NAO E' EM GERAL UTILIZADO NOS COMPILADORES, SENDO NESTE CASO DEIXADO AO PROGRAMADOR A RESPONSABILIDADE DE UTILIZAR CORRETAMENTE OS DESVIOS.

OS ERROS QUE SAO DETETADOS POR ESTE ALGORITMO SÃO DO SEGUINTE TIPO:

```
.....................................
   GO TO 30
.....................................
   DO 40 I = 1,10
.....................................
```

```
.....................................
30 WRITE (5,50) A
.....................................
40 CONTINUE
.....................................
```

OU:

```
.....................................
   DO 40 I = 1,10
.....................................
30 READ (8,60) B(I)
.....................................
40 CONTINUE
.....................................
   GO TO 30
.....................................
```

EM AMBOS OS CASOS, O DESVIO PARA O COMANDO 30 E' INVÁLIDO.

O ALGORITMO CONSISTE EM ASSOCIAR A CADA NUMERO DE COMANDO UM NÚMERO QUE REPRESENTA A SUA POSIÇÃO EM RELAÇÃO AOS COMANDOS "DO" (NAO UTILIZA A IDÉIA DE BLOCO). ESTES NÚMEROS SAO GERADOS SEQUENCIALMENTE, INCREMENTANDO-SE DE UM PARA CADA "DO" ENCONTRADO. COMPARANDO-SE OS NUMEROS ASSOCIADOS AOS NÚMEROS DE COMANDOS DURANTE AS SUAS DEFINIÇÕES E REFERÊNCIAS, E' POSSÍVEL DETERMINAR SE O DESVIO E' VALIDO OU NÃO (5).

8. REFERÊNCIAS A SÍMBOLOS AINDA NAO DEFINIDOS

DURANTE A COMPILAÇÃO, COMO SO E' POSSIVEL CONSULTAR O PROGRAMA FONTE UMA VEZ, EM VIRTUDE DO COMPILADOR SER DE UM PASSO APENAS, SURGEM INEVITÀVELMENTE PROBLEMAS COM AS REFERÊNCIAS A SIMBOLOS QUE AINDA NAO FORAM DEFINIDOS. UM MÉTODO PARA RESOLVER ISTO E' A CONSTRUÇÃO DE UM "VETOR DE TRANSFERENCIAS", EM QUE A REFERÊNCIA E' RESOLVIDA INDIRETAMENTE EM RELAÇÃO AO VETOR DE TRANSFERÊNCIAS, CUJO ENDEREÇO JA E' CONHECIDO NO MOMENTO DA COMPILACAO DA REFERENCIA. QUANDO O SIMBOLO E' DEFINIDO, BASTA COLOCAR O ENDEREÇO, AGORA JÁ CONHECIDO, NO LUGAR CORRETO DO VETOR DE TRANSFERÊNCIAS. ESTE PROCESSO E' SIMPLES, MAS O VETOR DE TRANSFERÊNCIAS OCUPA PALAVRAS DE MEMÓRIA.

UM OUTRO PROCESSO, QUE NÃO EMPREGA UM VETOR DE TRANSFERENCIAS, QUE FOI O UTILIZADO, SERA DESCRITO A SEGUIR. O PROCESSO CONSISTE EM UTILIZAR UMA LISTA DE APONTADORES, CONSTRUÍDA NAS PRÓPRIAS PALAVRAS EM QUE SERÃO POSTOS OS ENDEREÇOS DAS REFERENCIAS. PARA DESCREVER A IDÉIA, VAMOS ACOMPANHAR A COMPILAÇÃO DE COMANDOS QUE TENHAM REFERENCIAS A UM NUMERO DE COMANDO QUE AINDA NAO FOI DEFINIDO.

NA PRIMEIRA REFERÊNCIA AO COMANDO, E' CRIADA A ENTRADA NA TABELA DE SÍMBOLOS, QUE APONTARÁ PARA A REFERÊNCIA. A REFERÊNCIA, POR ENQUANTO CONTERÁ ZERO. O FLAG "1" INDICA QUE O ENDERECO E' UM PONTEIRO PARA A REFERÊNCIA, E NÃO UM ENDERECO REAL.

FORTRAN FONTE	SIMBÓLICO OBJETO	ENTRADA NA TABELA DE SIMBOLOS
.....	.....	
.....	.....	----------
GO TO 19	A B L 0	|19|A+1 |1|
.....	.....	----------
.....	.....	

NAS REFERÊNCIAS SEGUINTES, UMA LISTA E' CONSTRUÍDA, QUE COMECA NA TABELA DE SÍMBOLOS, CADA ELEMENTO DA LISTA E' UMA DAS REFERÊNCIAS, E APONTA PARA A REFERÊNCIA ANTERIOR. O FINAL DA LISTA, QUE CORRESPONDE A PRIMEIRA REFERÊNCIA, CONTEM ZEROS.

.....	.....	
GO TO 19	A B L 0	
.....	.....	----------
GO TO 19	B B L A+1	|19|C+1 |1|
.....	.....	----------
GO TO 19	C B L B+1	
.....	.....	

QUANDO O COMANDO FOR DEFINIDO, O ENDEREÇO DE MÁQUINA ASSOCIADO AO COMANDO 19 SERA CONHECIDO, E TÔDAS AS REFERÊNCIAS PODERÃO SER COMPLETADAS CAMINHANDO-SE NA LISTA, A PARTIR DA TABELA DE SÍMBOLOS. A LISTA E' NATURALMENTE DESFEITA (NEM E' MAIS NECESSÁRIA). SEJA "F" O ENDERECO ASSOCIADO AO COMANDO 19.

```
      ......          ............
    GO TO 19      A      B     L   F
      ......          ............
    GO TO 19      B      B     L   F        ----------
      ......          ............         |19|F   |0|
    GO TO 19      C      B     L   F        ----------
      ......          ............
19  CONTINUE      F      EQU       *
      ......          ............
```

NOTE-SE QUE O ENDERECO "F" E' TAMBEM POSTO NA TABELA DE SÍMBOLOS, DE TAL MODO QUE DE AGORA EM DIANTE, QUALQUER REFERÊNCIA AO COMANDO 19 PODE SER RESOLVIDA IMEDIATAMENTE. ISTO E' INDICADO TROCANDO-SE O FLAG "1" POR "0".

ESTE ALGORITMO E' TAMBEM UTILIZADO PARA RESOLVER AS REFERÊNCIAS A SUBPROGRAMAS AINDA NAO DEFINIDOS.

9. CONVERSÕES BINARIAS-DECIMAIS

COMO O COMPUTADOR IBM-1130 E' BINÁRIO, SÃO NECESSÁRIAS CONVERSÕES DECIMAIS-BINARIAS DE CONSTANTES DURANTE A COMPILAÇÃO E DURANTE A LEITURA DE DADOS NA EXECUÇÃO. SÃO NECESSÁRIAS TAMBEM CONVERSÕES BINÁRIAS-DECIMAIS PARA A IMPRESSÃO DE RESULTADOS, DURANTE A EXECUÇÃO. ESTAS CONVERSÕES PARA NÚMEROS REAIS, QUE ENVOLVE MUDANÇAS DE NÚMEROS COM MANTISSAS E EXPOENTES DECIMAIS PARA NÚMEROS EQUIVALENTES COM MANTISSAS E EXPOENTES BINÁRIOS, ESTÃO LONGE DE SEREM TRIVIAIS.

NA BIBLIOTECA DE PROGRAMAS DO MONITOR, HÁ UMA SUBROTINA COM AS FUNÇÕES DESCRITAS ACIMA, MAS ESTA E' BEM LENTA, POIS EMPREGA UM PROCESSO ITERATIVO, EM QUE SÃO NECESSÁRIAS MUITAS DEZENAS DE ITERAÇÕES. COMO O OBJETIVO PRIMORDIAL DO COPPE-FORTRAN E' A VELOCIDADE, DECIDIMOS ELABORAR UMA OUTRA ROTINA, CUJO ALGORITMO UTILIZA UMA TABELA DE ACESSO DIRETO, CONTENDO DIVERSAS POTENCIAS DE DEZ, EM NOTAÇÃO BINÁRIA. DESTE MODO, CONSEGUIMOS REDUZIR O TEMPO DAS CONVERSÕES DE 50 MILISEGUNDOS PARA 10 MILISEGUNDOS, EM MEDIA.

10. DIAGNÓSTICOS

UM PONTO QUE CERTAMENTE MERECE CUIDADOS ESPECIAIS, E' A EMISSÃO DE DIAGNÓSTICOS DURANTE A COMPILAÇÃO, JÁ QUE DIAGNÓSTICOS DETALHADOS E' UMA DAS METAS DO SISTEMA.

NESTE PONTO, O FATO DO COMPILADOR SER DE PASSO ÚNICO VEM FACILITAR O NOSSO TRABALHO, VISTO QUE PODEREMOS INDICAR O ERRO AO LADO DO COMANDO ERRADO NA LISTAGEM, AO INVÉS DE INDICÁ-LO NO FINAL DA LISTAGEM, COMO FAZ O MONITOR. PODEMOS FAZER MAIS AINDA: PODEMOS, POR INTERMÉDIO DE UM PONTEIRO, INDICAR A REGIÃO PROVÁVEL DE ERRO, JA QUE O ANALISADOR DE SINTAXE PODE-NOS DAR ESTA INFORMAÇÃO.

PARA APRIMORAR AINDA MAIS ESTE ESQUEMA, PODEMOS IMPRIMIR O TEXTO DAS MENSAGENS DE ÊRRO NO FINAL DA LISTAGEM DO PROGRAMA, DESDE QUE SEJA PROVIDA UMA ÁREA ONDE SERÃO ARMAZENADOS OS TIPOS DE ERROS DETETADOS DURANTE A COMPILAÇÃO.

FINALMENTE, PODEMOS, PARA AUXILIAR O PROGRAMADOR PRINCIPIANTE, TENTAR EXECUTAR PROGRAMAS COM ERROS "LEVES", DESIGNANDO CADA SINTAXE INVÁLIDA DE "ÉRRO" OU "ADVERTÊNCIA". OS ERROS SERIAM AS IMPERFEIÇÕES MAIS GRAVES, E IMPEDIRIAM A EXECUÇÃO, ENQUANTO QUE AS ADVERTÊNCIAS SERIAM AS MENOS GRAVES, QUE NAO IMPEDIRIAM A EXECUCAO, COMO POR EXEMPLO ESQUECER DE NUMERAR UM COMANDO SEGUINTE A UM COMANDO DE DESVIO. O QUE NAO PODE SER ESQUECIDO E' QUE UMA IMPERFEIÇÃO DESIGNADA COMO "ADVERTENCIA" NAO IRÁ IMPEDIR A EXECUÇÃO DO PROGRAMA, E PORTANTO NESTES CASOS AINDA O COMPILADOR TERÁ DE GERAR UM PROGRAMA OBJETO EXECUTÁVEL.

```
**************************************
*                                    *
* CAPITULO VI   *   MACRO-ASSEMBLER  *
*                                    *
**************************************
```

APROXIMADAMENTE UM ANO APÓS O INICÍO DA ELABORAÇÃO DO SISTEMA, FOI LANÇADO, PELA IBM, O MACRO-ASSEMBLER PARA O 1130. COMPARADO COM O ASSEMBLER COMUM, O MACRO-ASSEMBLER E' UM INSTRUMENTO MUITO MAIS PODEROSO, POIS PERMITE DEFINIR UMA SÉRIE DE MACRO-INSTRUÇÕES GUARDADAS EM DISCO, QUE PODEM SER REFERIDAS EM UM PROGRAMA DE LINGUAGEM SIMBÓLICA QUALQUER (12). ELE CERTAMENTE SERÁ ÚTIL PARA O DESENVOLVIMENTO DO SISTEMA.

A PRIMEIRA APLICAÇÃO PARA O MACRO-ASSEMBLER E' A DE PERMITIR RECUPERAR NOVAMENTE A FACILIDADE DE REALIZAR MODIFICAÇÕES, PERDIDA AO DECIDIR-SE MONTAR TODAS AS ROTINAS JUNTAS. ISTO E' CONSEGUIDO DA SEGUINTE MANEIRA: EM UMA BIBLIOTECA DE MACROS, CADA SUBROTINA E' GUARDADA COMO UMA MACRO. A FASE, QUE E' UM CONJUNTO DE SUBROTINAS MONTADAS JUNTAS, E' GUARDADA TAMBEM COMO UMA MACRO, QUE CONTEM AS REFERENCIAS AS SUBROTINAS QUE O COMPÕE (QUE SAO MACROS). PORTANTO, PARA SE REALIZAR A MONTAGEM DE UMA FASE E' NECESSÁRIO PROCESSAR UM PROGRAMA EM LINGUAGEM SIMBÓLICA CONTENDO APENAS UM CARTÃO (ALEM DO "END") - O CARTAO QUE CONTEM A REFERÊNCIA A MACRO DO MÓDULO, E AS MACROS DAS SUBROTINAS SERAO EXPANDIDAS, GERANDO O MÓDULO. UM EXEMPLO DO PROGRAMA PARA O MONTAR O MODULO "$RESI" SERIA:

```
$RESI
END     $RESI
```

PARA MODIFICAR UMA DAS SUBROTINAS, BASTA MODIFICÁ-LA NA BIBLIOTECA DE MACROS, E REMONTAR O SISTEMA. NOTE-SE QUE ISTO ENVOLVE APENAS A LEITURA FISICA DA NOVA VERSÃO DA SUBROTINA, AO INVÉS DA LEITURA DE TODO O MODULO, NECESSARIO PELO METODO ANTERIOR (FIG. 10).

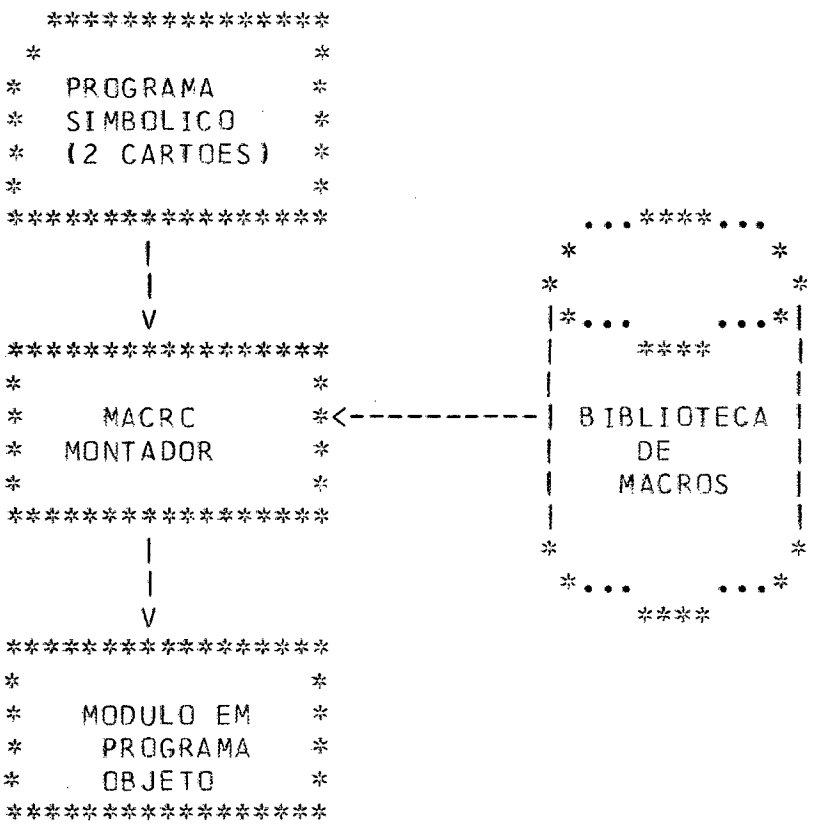


FIG. 10: MONTANDO A FASE ATRAVES DO "MACRO-ASSEMBLER"

O USO DO MACRO-ASSEMBLER PERMITE MAIS SOFISTICAÇÕES: COM O AUXÍLIO DA MONTAGEM CONDICIONAL, E' RELATIVAMENTE SIMPLES FAZER COM QUE O SISTEMA CONTENHA TODAS AS MENSAGENS EM PORTUGUES E INGLES (OU OUTRA LINGUA QUALQUER), DAS QUAIS UMA PODERÁ SER ESCOLHIDA DURANTE A MONTAGEM, MUDANDO-SE APENAS UM PARÂMETRO. DESTE MODO E' POSSIVEL OBTER-SE VERSÕES EM VÁRIAS LÍNGUAS DO SISTEMA.

PARA A REALIZAÇÃO PRATICA, UTILIZOU-SE O SEGUINTE ESQUEMA: DOS TRES "DRIVES" DISPONIVEIS, USOU-SE UM "DRIVE" PARA CONTER UM DOS DISCOS DA BIBLIOTECA DE MACROS (FORAM NECESSÁRIOS AO TOTAL 2 DISCOS PARA CONTER AS MACROS DE TODAS AS FASES), O SEGUNDO PARA CONTER A ÁREA DE TRABALHO EM QUE FICARIA A FASE EM FORMA EXPANDIDA, E O TERCEIRO "DRIVE" PARA CONTER O DISCO COM O MONITOR E O MACRO-ASSEMBLER. COM ESTA CONFIGURAÇÃO, QUE PERMITE A MAIOR VELOCIDADE, O TEMPO DE MONTAGEM DE UMA FASE VARIA ENTRE 10 E 45 MINUTOS, SEM A LISTAGEM DO PROGRAMA-FONTE.

COM MAIS "DRIVES" DISPONÍVEIS, NAO HAVERÁ AUMENTO DA VELOCIDADE DE MONTAGEM, POIS CADA ARQUIVO EM DISCO JÁ ESTÁ OCUPANDO SEU "DRIVE" SEPARADO. COM MENOS "DRIVES" DISPONÍVEIS, TAMBEM E' POSSIVEL UTILIZAR O ESQUEMA DE MACROS. COM 2 DRIVES, PODEMOS POR A AREA DE TRABALHO NO DISCO QUE CONTEM O MONITOR. COM 1 "DRIVE" APENAS, PODEMOS POR TODOS OS ARQUIVOS EM UM DISCO APENAS, DESDE QUE DISPONHAMOS DE VÁRIOS DISCOS, UM PARA A MONTAGEM DE CADA FASE.

O INÍCIO DA UTILIZAÇÃO DO MACRO-ASSEMBLER FOI SEM DÚVIDA UM MARCO MUITO IMPORTANTE NO DESENVOLVIMENTO DO SISTEMA. SEM ÊLE, TERÍAMOS DE CONTINUAR UTILIZANDO O MÉTODO ANTERIOR DE MONTAGEM, EM QUE ERA NECESSÁRIO SEMPRE LER PELA LEITORA DE CARTÕES TODAS AS SUBROTINAS QUE FORMAM CADA FASE.

CAPITULO VII * PROGRAMA OBJETO

NESTE CAPITULO IREMOS DISCUTIR OS CRITÉRIOS UTILIZADOS PARA A ESTRUTURAÇÃO DO PROGRAMA OBJETO GERADO PELO COMPILADOR DO SISTEMA.

INICIALMENTE VAMOS DESCREVER SUSCINTAMENTE O FUNCIONAMENTO DO COMPILADOR FORTRAN DO MONITOR. NESTE COMPILADOR, O PROGRAMA FONTE E' INICIALMENTE LIDO, E COLOCADO INTEGRALMENTE NA MEMORIA INTERNA EM UM FORMATO COMPACTADO. NESTA ETAPA, SE DESEJADO, O PROGRAMA FONTE E' LISTADO.

A MEMÓRIA E' DIVIDIDA BASICAMENTE EM DUAS PARTES: A AREA DE "OVERLAYS", DE 1500 PALAVRAS, EM QUE AS 28 FASES DO COMPILADOR VAO SENDO LIDAS DO DISCO, E O RESTANTE, QUE E' UTILIZADO PARA O PROGRAMA FONTE. CADA FASE (QUE REALIZA UM PASSO) DO COMPILADOR PROCESSA O PROGRAMA DADO, TRANSFORMANDO-O GRADUALMENTE NO PROGRAMA OBJETO. AO FINAL DA ULTIMA FASE, TEMOS PRODUZIDO O PROGRAMA OBJETO, QUE E' ENTAO COPIADO PARA A ÁREA DE TRABALHO DO DISCO, DE ONDE SERA MAIS TARDE RECUPERADO.

DESCREVEREMOS A SEGUIR, O PROGRAMA OBJETO GERADO PELO COMPILADOR FORTRAN DO MONITOR. ELE SE COMPÕE DE UM CONJUNTO DE CHAMADAS A SUBPROGRAMAS, QUE IRÃO ESTAR PRESENTES DURANTE A EXECUÇÃO, SEGUIDO DE ARGUMENTOS PARA ESTES SUBPROGRAMAS.

EX.: PARA O COMANDO ARITMETICO

```
A = B
```

E' CRIADO O PROGRAMA OBJETO

```
LIBF    FLD
DC      B
LIBF    FSTC
DC      A
```

A PSEUDO-OPERAÇÃO "LIBF" E' UMA REFERÊNCIA A UM SUBPROGRAMA (NO CASO "FLD" OU "FSTO"), QUE CAUSA A TRANSFERÊNCIA DO CONTRÔLE PARA ESTE, E PROVE O ENDERECO DE RETORNO. O ENDEREÇO DO ARGUMENTO, "B" VEM LOGO EM SEGUIDA. O OBJETIVO DO SUBPROGRAMA NO CASO, E' CARREGAR O VALOR DA VARIAVEL "B" NO ACUMULADOR FLUTUANTE. AS DUAS INSTRUÇOES QUE VEM A SEGUIR SEGUEM A MESMA ESTRUTURA, E TEM COMO OBJETIVO ATRIBUIR O VALOR DO ACUMULADOR FLUTUANTE PARA A VARIÁVEL "A", OBTENDO DESTE MODO O EFEITO DESEJADO PARA O COMANDO FORTRAN DADO. O PROGRAMA OBJETO DO MONITOR SEGUE ESTA ESTRUTURA NA QUASE TOTALIDADE, POIS EMPREGA SUBPROGRAMAS PARA REALIZAR QUASE TODAS AS FUNÇÕES DESEJADAS. UMA EXCESSÃO A ISTO SAO AS OPERAÇÕES ARITMÉTICAS ENVOLVENDO APENAS OPERANDOS INTEIROS, EM QUE SAO UTILIZADAS AS PRÓPRIAS INSTRUÇÕES DE MÁQUINA.

EXEMPLO: DADO O COMANDO ARITMETICO

```
I = J + K
```

O PROGRAMA OBJETO GERADO E':

```
LD    L  J
A     L  K
STO   L  I
```

AS INSTRUÇÕES ACIMA TEM A FUNÇÃO DE CARREGAR O VALOR DE "J" NO ACUMULADOR DA UNIDADE CENTRAL DE PROCESSAMENTO, SOMAR A ESTE ACUMULADOR O VALOR DE "K", E FINALMENTE ATRIBUIR O RESULTADO A VARIAVEL "I".

O PROJETO DO PROGRAMA OBJETO DO MONITOR ENVOLVEU O COMPROMISSO ENTRE BOA VELOCIDADE DE EXECUCAO E POUCA MEMÓRIA OCUPADA. ISTO PORQUE, COMO O 1130 NAO TEM CIRCUITO PARA ARITMÉTICA DE PONTO FLUTUANTE E' NECESSARIO PROVER UMA SIMULACAO PARA ELA, E NATURALMENTE FAZER ISTO COM SUBROTINAS OCUPA MUITO MENOS MEMÓRIA DO QUE INCORPORAR AS INSTRUÇÕES NECESSÁRIAS AO PROGRAMA OBJETO, TODA VEZ QUE NECESSÁRIO.

POR OUTRO LADO, COMO O COMPUTADOR JA POSSUI INSTRUÇÕES PARA REALIZAR AS OPERACOES INTEIRAS, ESTAS SAO UTILIZADAS DURANTE O PROGRAMA OBJETO SEMPRE QUE SÃO NECESSÁRIAS, PROVENDO VELOCIDADE DE EXECUÇÃO RÁPIDA, BEM SUPERIOR A CONSEGUIDA POR UMA SUBROTINA, E OCUPANDO O MESMO NUMERO DE PALAVRAS DE MEMÓRIA DO QUE AS REFERÊNCIAS AS SUBROTINAS.

PARA O COPPE-FORTRAN, COMO AOS OBJETIVOS DE PROGRAMA

OBJETO COMPACTO E VELOCIDADE RAZOÁVEL DE EXECUÇÃO SE ACRESCENTA AINDA O DE PROVER BONS DIAGNOSTICOS DURANTE A EXECUÇÃO, SERÃO NECESSÁRIAS ALGUMAS MODIFICAÇÕES NA ESTRUTURA DO PROGRAMA OBJETO.

A UTILIZAÇÃO DE REFERÊNCIAS AS SUBROTINAS, COM O EMPREGO DE ENDEREÇOS DOS ARGUMENTOS E' UM IDÉIA QUE DEVE SER APROVEITADA, VISTO QUE GERAR TODAS AS INSTRUÇÕES SEMPRE QUE NECESSÁRIAS TORNARIA O TAMANHO DO PROGRAMA OBJETO PROIBITIVO.

PARA PROVER BONS DIAGNOSTICOS DURANTE A EXECUÇÃO TORNA-SE NECESSÁRIO UTILIZAR SUBROTINAS MESMO PARA FUNÇÕES PARA AS QUAIS EXISTEM INSTRUÇÕES DE MÁQUINA. ISTO SE DEVE AO FATO DE QUE ESTAS INSTRUÇÕES DE MÁQUINA NÃO ACUSAM AUTOMATICAMENTE O ERRO (COMO POR EXEMPLO, ATRAVÉS DE UMA INTERRUPÇÃO). UM EXEMPLO BOM PARA ESTE FATO E' A EXECUÇÃO DO "GO TO" COMPUTADO.

```
GO TO (10,20,30,40),K
```

ISTO PODERIA SER COMPILADO APENAS COM INSTRUCOES DE MÁQUINA, COMO POR EXEMPLO

```
      LDX   I1 K
      B     I1 ADDR-1
ADDR  DC       (10) END. DO COM. 10
      DC       (20) END. DO COM. 20
      DC       (30) END. DO COM. 30
      DC       (40) END. DO COM. 40
```

MAS SE AO EXECUTARMOS O PROGRAMA O VALOR DE "K" FOR POR EXEMPLO, 5, O RESULTADO SERA IMPREVISÍVEL, PROVAVELMENTE CAUSANDO O COLAPSO DO SISTEMA.

A SOLUÇÃO NESTE CASO, E' COMPILAR O COMANDO ATRAVES DA CHAMADA A UMA SUBROTINA QUE FAÇA A CONSISTENCIA DO VALOR DE "K", COMO POR EXEMPLO:

```
      LIBF     XGOTO
      DC       K
      DC       4    VALOR MAXIMO DE K
      DC       (10) END. DO COMANDO 10
      DC       (20) END. DO COMANDO 20
      DC       (30) END. DO COMANDO 30
      DC       (40) END. DO COMANDO 40
```

A SUBROTINA "XGOTO" RECEBERÁ O VALOR DE "K", VERIFICARÁ SE O SEU VALOR ESTA DENTRO DOS LIMITES PERMITIDOS (1 A 4) E FARIA O DESVIO ADEQUADO ATRAVES DA TABELA DE ENDEREÇOS. EM CASO DE ERRO, TERIA TODAS AS CONDIÇÕES PARA FORNCECER UM DIAGNÓSTICO BEM PRECISO.

O TEMPO DE EXECUÇÃO PARA O EXEMPLO IRÁ AUMENTAR UM POUCO, VISTO QUE E' NECESSARIO ANALISAR O VALOR DE "K", E REALIZAR A LIGAÇÃO COM A SUBROTINA "XGOTO", COISAS QUE NAO SÃO NECESSÁRIAS PARA O PRIMEIRO PROGRAMA DADO.

UTILIZANDO ESTE ESQUEMA, MANTEMOS APROXIMADAMENTE O GASTO DE MEMÓRIA (NO CASO DADO ATE ABAIXOU DE 8 PARA 7 PALAVRAS), AUMENTAMOS LIGEIRAMENTE O TEMPO DE EXECUÇÃO, E GANHAMOS MUITO EM TERMOS DE DIAGNÓSTICOS.

ESTA FILOSOFIA FOI A UTILIZADA PARA TODO O PROGRAMA OBJETO, POIS PROVÊ BONS COMPROMISSOS ENTRE MEMÓRIA OCUPADA, VELOCIDADE DE EXECUÇÃO E A EMISSÃO DE BONS DIAGNÓSTICOS.

PARA PODERMOS ASSOCIAR UM ÊRRO DE EXECUÇÃO AO COMANDO FORTRAN QUE O GEROU, DECIDIMOS INCORPORAR AO PROGRAMA OBJETO, ANTES DO CÓDIGO GERADO POR CERTO COMANDO FORTRAN, UMA CHAMADA A UMA ROTINA DE CONTRÔLE SEGUIDO DO NÚMERO DA LINHA (ISN) DO COMANDO FORTRAN. EXEMPLO:

PARA O COMANDO FORTRAN

```
GO TO 38
```

SERIA GERADO O CODIGO

```
LIBF    CONTR     DESVIA P/ "CONTR"
DC      53        NUMERO DA LINHA
B    L  (38)      DESVIA P/ COM. 38
```

ONDE 53 E' O NUMERO DA LINHA, ASSOCIADO PELO COMPILADOR AO COMANDO "GO TO", IMPRESSO NA LISTAGEM AO LADO DO COMANDO. DURANTE A EXECUÇÃO, A ROTINA DE CONTROLE "CONTR" GUARDA O NUMERO 53 PARA O CASO DE ERROS, ALEM DE ATUALIZAR UM CONTADOR, DESTINADO A INTERROMPER AUTOMATICAMENTE PROGRAMAS EM "LOOP".

PARA VERIFICARMOS SE O PROGRAMA OBJETO ESTA SENDO GERADO CORRETAMENTE PELO COMPILADOR, DECIDIMOS ELABORAR UM PROGRAMA COM A FINALIDADE DE LISTAR O PROGRAMA OBJETO EM FORMATO SIMBÓLICO, SEMELHANTE A LINGUAGEM ASSEMBLER, UTILIZANDO OS NOMES DAS PRÓPRIAS VARIÁVEIS UTILIZADAS NO PROGRAMA EM FORTRAN. ISTO E' CONSEGUIDO A PARTIR DO PROGRAMA OBJETO, EM LINGUAGEM DE MÁQUINA, QUE RESIDE NA MEMORIA INTERNA.

CAPITULO VIII * ALOCAÇÃO DE MEMORIA

EM RELAÇÃO A DISTRIBUIÇÃO DE MEMÓRIA AOS DIVERSOS INTEGRANTES DO SISTEMA, UMA SÉRIE DE COMPROMISSOS DEVEM SER MANTIDOS. O PRINCIPAL DESTES E' O QUE OCORRE ENTRE A VELOCIDADE DE PROCESSAMENTO E O TOTAL DE MEMÓRIA DISPONÍVEL PARA O PROGRAMA OBJETO. QUANTO MAIS MEMÓRIA O SISTEMA OCUPA, MAIOR A EFICIENCIA, MAS EM COMPENSAÇÃO MENOR SERA A MEMÓRIA DISPONÍVEL PARA O PROGRAMA OBJETO, E EM CONSEQUENCIA MENOR O PROGRAMA QUE PODERÁ SER PROCESSADO.

PARA INICIAR A ALOCAÇÃO DE MEMORIA, PODEMOS ALOCAR OS COMPONENTES QUE POR DECISÕES ANTERIORES, SABEMOS QUE DEVERÃO SER SEMPRE RESIDENTES. ENTRE ESTES COMPONENTES TEMOS O MONITOR RESIDENTE E A PARTE RESIDENTE DO SISTEMA (QUE CONTEM A AREA DE COMUNICAÇÕES - FIG. 11).

FIG. 11: MEMÓRIA TOTAL DISPONÍVEL,
JA COM A PARTE RESIDENTE INDICADA

A MEMÓRIA QUE RESTA DEVERA SER DIVIDIDA ENTRE AS 3 FASES DO SISTEMA, E A REGIÃO DISPONÍVEL PARA O PROGRAMA OBJETO. NESTE PONTO DURANTE O DESENVOLVIMENTO DO SISTEMA, NAO CONHECEMOS O TAMANHO DE TODOS OS MÓDULOS QUE O COMPÕEM, E PORTANTO NAO SE PODE REALIZAR A ALOCAÇÃO COM CRITÉRIO. DECIDIMOS PORTANTO PROVISORIAMENTE, ALOCAR OS VARIOS INTEGRANTES SEQUENCIALMENTE (FIG. 12).

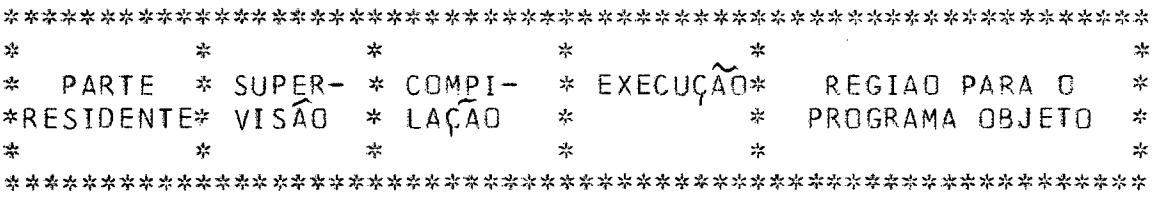


FIG. 12: MEMÓRIA TOTAL, INDICANDO AS DIVERSAS FASES

DEPOIS DE TERMINADAS TODAS AS 3 FASES, PUDEMOS VERIFICAR QUE TORNANDO RESIDENTES ESTAS FASES, SOBRARIAM CERCA DE 8000 PALAVRAS PARA A REGIÃO DO PROGRAMA OBJETO, O QUE E' BEM RAZOÁVEL, POIS PERMITE O PROCESSAMENTO DE PROGRAMAS DE ATÉ CERCA DE 400 CARTOES (SUPONDO NATURALMENTE, QUE OS VETORES E MATRIZES SEJAM PEQUENOS). COMO PODEMOS VER, ESTA IDEIA NAO PROVE ESPAÇO PARA OS MÓDULOS QUE IRÃO IMPRIMIR AS MENSAGENS DE ERRO, A TABELA DE SÍMBOLOS E O PROGRAMA OBJETO. PARA TAL, TORNOU-SE NECESSÁRIO CONSTRUIR UM "OVERLAY". DECIDIU-SE EFETUAR O "OVERLAY" NA ÁREA OCUPADA PELA FASE DE EXECUÇÃO, VISTO QUE ELA E' A FASE MENOS UTILIZADA, EM VIRTUDE DO ALTO NÚMERO DE ERROS DE COMPILAÇÃO DOS PROGRAMAS SIMPLES (FIG. 13).

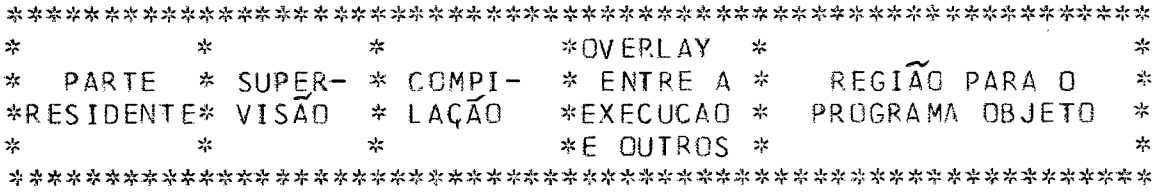


FIG. 13: MEMÓRIA TOTAL, INDICANDO
A ÁREA DE OVERLAYS

DENTRO DA REGIÃO DISPONÍVEL PARA O PROGRAMA OBJETO, TEMOS DE ALOCAR DUAS PARTES QUE TEM CRESCIMENTO INDEPENDENTE: O CÓDIGO DO PROGRAMA OBJETO E OS DADOS DO PROGRAMA OBJETO. PARA APROVEITAR AO MAXIMO A REGIÃO, DECIDIMOS FAZER O CÓDIGO COMEÇAR DO LIMITE INFERIOR EM SENTIDO CRESCENTE, E OS DADOS A PARTIR DO LIMITE SUPERIOR, EM SENTIDO DECRESCENTE (FIG. 14).

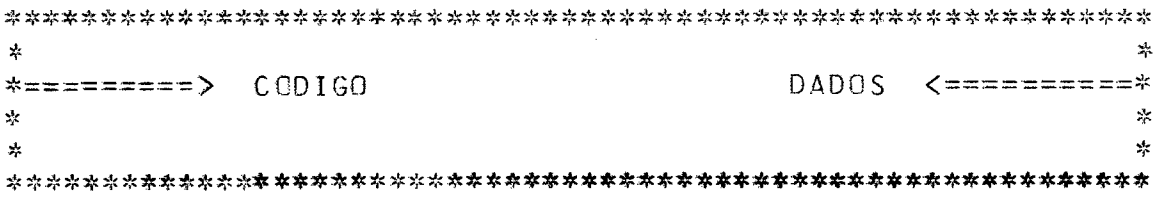


FIG. 14: REGIÃO DO PROGRAMA OBJETO
(ESCALA AMPLIADA)

DESTE MODO, QUANDO AS DUAS PARTES COM SENTIDOS CONTRÁRIOS SE ENCONTRAM, A REGIÃO ESTARÁ TOTALMENTE ESGOTADA. A EXTENSÃO, NO CASO DE MAIS DE UM PROGRAMA FONTE, E' SIMPLES (FIG. 15).

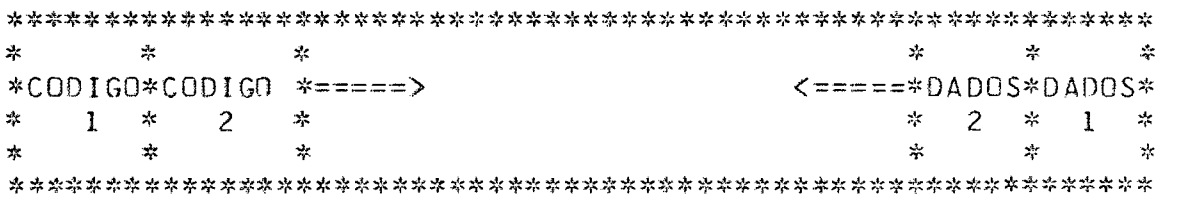


FIG. 15: REGIÃO DO PROGRAMA OBJETO
PARA VÁRIOS PROGRAMAS

PARA O EMPREGO DO SISTEMA COM 16K, FOI NECESSÁRIO EMPREGAR UM ESQUEMA MAIS COMPLEXO DE "OVERLAYS", QUE NATURALMENTE IRÁ BAIXAR A EFICIÊNCIA DO SISTEMA, MAS SEM RETIRAR NENHUMA FUNÇÃO EFETUADA PELO MESMO.

PARA ESTE ESQUEMA, APENAS FICOU RESIDENTE NA MEMORIA O MONITOR RESIDENTE, A ÁREA DE COMUNICAÇÕES, E O "FLIPPER" PARA REALIZAR OS "OVERLAYS". ESTE "FLIPPER" E' UMA ROTINA QUE CONTROLA OS "OVERLAYS", LENDO OS SEGMENTOS DOS PROGRAMAS NECESSÁRIOS. TODOS OS DEMAIS INTEGRANTES FORAM POSTOS EM ÁREAS TRANSIENTES, UTILIZANDO-SE 3 NÓS (FIG. 16).

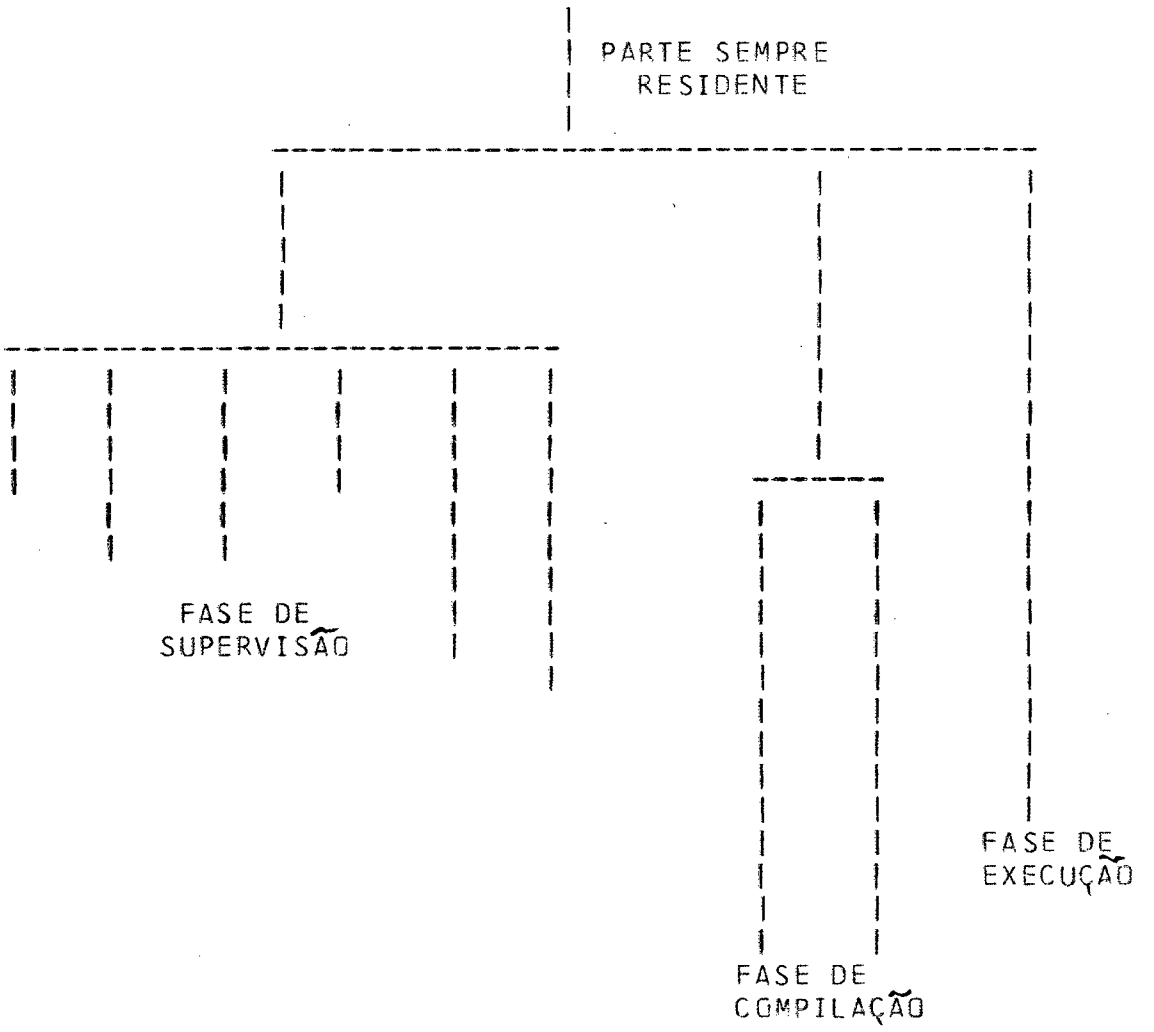


FIG. 16: MEMÓRIA DISPONÍVEL (EXCLUÍDA A REGIÃO DO PROGRAMA OBJETO) NA VERSAO DE 16K, INDICANDO OS "OVERLAYS" REALIZADOS.

CAPITULO IX * FASE DE EXECUÇÃO

COMO FOI VISTO NO CAPITULO VII, O PROGRAMA OBJETO CONSISTE BASICAMENTE DE UMA SÉRIE DE CHAMADAS A SUBROTINAS, QUE ESTÃO PRESENTES NA MEMÓRIA DURANTE A EXECUÇÃO DO PROGRAMA. A FASE DE EXECUÇÃO CONSISTE JUSTAMENTE DESTAS SUBROTINAS. ESTAS SUBROTINAS PODEM SER DIVIDIDAS EM 5 GRUPOS, DE ACORDO COM AS SUAS FINALIDADES:

1. SUBROTINAS ARITMÉTICAS: SAO AS QUE EFETUAM AS OPERAÇÕES DE ADIÇÃO, SUBTRAÇÃO, DIVISÃO, MULTIPLICAÇÃO E POTENCIAÇÃO DE NÚMEROS REAIS E INTEIROS. ELAS SÃO NECESSÁRIAS, POIS O 1130 NAO POSSUI CIRCUITO PARA AS OPERAÇÕES DE PONTO FLUTUANTE E NAO PRODUZ INTERRUPÇÕES PARA ERROS OCORRIDOS DURANTE AS OPERAÇÕES INTEIRAS. PARA A CONFECÇÃO DESTAS SUBROTINAS, NOS BASEAMOS NAS SUBROTINAS DO MONITOR, QUE REALIZAM AS MESMAS FUNÇÕES, PARA EVITAR DUPLICAÇÃO DE ESFORÇOS. TIVERAM NO ENTANTO, DE SEREM ADAPTADAS A DIFERENTE FILOSOFIA DE LIGAÇÃO E TRATAMENTO DE ERROS.

2. SUBROTINAS FUNCIONAIS: SAO AS QUE EFETUAM O CALCULO DAS FUNÇÕES E SUBROTINAS PADRÕES DA LINGUAGEM, TAIS COMO O "SIN", "ALOG", "EXP", ETC... COMO OCORREU COM AS SUBROTINAS ARITMÉTICAS, NOS BASEAMOS NAS SUBROTINAS DO MONITOR, EFETUANDO AS NECESSÁRIAS MODIFICAÇÕES.

3. SUBROTINAS DE ENTRADA/SAIDA: SAO AS SUBROTINAS QUE PROCESSAM OS COMANDOS "READ" E "WRITE", EFETUANDO AS CONVERSÕES NECESSÁRIAS PARA ARMAR UMA LINHA DE IMPRESSÃO OU DECOMPOR UM CARTÃO DE DADOS. ESTE GRUPO DE SUBROTINAS NAO PODE SER BASEADO NAS ROTINAS DE ENTRADA/SAIDA DO MONITOR, POR QUE ESTAS UTILIZAM UM CÓDIGO DE CARACTERES E TRATAMENTO DE ERROS DIFERENTES DO DESEJADO, E NÃO POSSUEM ENTRADA/SAIDA SEM FORMATO.

4. SUBROTINAS PARA A EXECUÇÃO DE COMANDOS ESPECIAIS: SÃO SUBROTINAS QUE "INTERPRETAM" CERTOS COMANDOS, TAIS COMO O "GO TO" COMPUTADO, O FINAL DA MALHA DO "DO" E O COMANDO "IF". A NECESSIDADE DESTAS SUBROTINAS DECORRE PRINCIPALMENTE DO OBJETIVO DE PROVER BONS DIAGNOSTICOS, VISTO QUE PODERÍAMOS PRODUZIR PROGRAMAS OBJETOS PARA TAIS COMANDOS QUE NAO CONTIVESSEM REFERÊNCIAS A SUBROTINAS. COMO O MONITOR NAO POSSUI TAIS SUBROTINAS, ELAS TIVERAM DE SER FEITAS ESPECIALMENTE.

5. SUBROTINAS DE SERVIÇO E AUXILIARES: ESTE GRUPO CONTÉM UM GRANDE NUMERO DE SUBROTINAS AUXILIARES QUE SAO UTILIZADAS PELAS SUBROTINAS DOS GRUPOS ANTERIORES, NAO SENDO REFERENCIADAS DIRETAMENTE NO PROGRAMA OBJETO. ENTRE ELAS TEMOS ROTINAS DE CONVERSÃO DE CÓDIGOS, CONVERSÃO BINÁRIA/DECIMAL E TRATAMENTO DE ERROS. ESTAS ROTINAS FORAM FEITAS INDEPENDENTES PRINCIPALMENTE POR CONSIDERAÇÕES DE MODULARIDADE.

AS SUBROTINAS DOS GRUPOS 1 E 2 FORAM BASEADAS EM SUBROTINAS ANÁLOGAS DO MONITOR, ENQUANTO QUE AS DEMAIS (GRUPOS 3, 4 E 5) SAO ORIGINAIS, DEVIDO A AUSÊNCIA DE SUBROTINAS ANÁLOGAS DO MONITOR, OU PORQUE ESTAS DIFERIAM MUITO DO DESEJADO.

A TÉCNICA DE LIGAÇÃO ENTRE O PROGRAMA OBJETO E OS DIVERSOS SUBPROGRAMAS UTILIZADOS PODE SER DE DOIS TIPOS: O "LIBF", CUJA REFERENCIA NECESSITA DE APENAS UMA PALAVRA, MAS NECESSITA DE UM VETOR DE TRANSFERÊNCIA, E O "CALL", CUJA REFERÊNCIA OCUPA 2 PALAVRAS, MAS NAO NECESSITA DO VETOR DE TRANSFERENCIA.

PARA PROVER O NIVEL DE DIAGNÓSTICOS DESEJADO PELAS PREMISSAS DO SISTEMA, AS ROTINAS DE EXECUÇÃO ACIMA MENCIONADAS REALIZAM UMA CONSISTENCIA EM TODOS OS ARGUMENTOS FORNECIDOS NA RESPECTIVA CHAMADA PELO PROGRAMA OBJETO, E IMPRIMEM UMA MENSAGEM DE ERRO DETALHADA A CADA ERRO DETETADO, FORNECENDO MEIOS PARA QUE O PROGRAMADOR ENCONTRE A CAUSA DO ERRO. ASSIM, PARA CADA ERRO E' IMPRESSO O NUMERO DA LINHA DO PROGRAMA FONTE QUE ESTAVA SENDO EXECUTADO, QUANDO O ERRO FOI DETETADO. ISTO E' POSSIVEL EMPREGANDO-SE O NUMERO DA LINHA QUE E' INCORPORADO AO PROGRAMA OBJETO. DEPENDENDO DO ERRO, MAIS INFORMAÇÕES PODEM SER DADAS, COMO POR EXEMPLO, O ENDEREÇO DE ALGUMA VARIAVEL, VALOR DE ALGUMA VARIAVEL SIGNIFICATIVA, ETC...

A DETEÇÃO DE MUITOS DESTES ERROS, COMO POR EXEMPLO, O USO DE ÍNDICES DE VARIÁVEIS SUBSCRITAS FORA DOS LIMITES,

AUXILIA NÃO SOMENTE O PROGRAMADOR, MAS TAMBÉM O SERVIÇO DE OPERAÇÃO, VISTO QUE TAIS ERROS CAUSAM PARADAS INVÁLIDAS DO COMPUTADOR, DEVIDO A DESTRUIÇÃO DO SUPERVISOR RESIDENTE, QUANDO O PROGRAMA E' PROCESSADO PELO MONITOR.

AS ROTINAS DE EXECUÇÃO, TANTO NA VERSAO DE 32K, COMO NA DE 16K, RESIDEM EM UMA AREA DE "OVERLAYS". AO SER INICIADA A EXECUÇÃO, AS ROTINAS SAO CARREGADAS (SE JÁ NÃO ESTAO RESIDENTES), INTEGRALMENTE, INDEPENDENTEMENTE DE QUAIS OU QUANTAS SAO EFETIVAMENTE UTILIZADAS.

A ORDEM USADA NAS SUBROTINAS NO CONJUNTO FOI A SEGUINTE: EM PRIMEIRO LUGAR AS SUBROTINAS FUNCIONAIS (GRUPO 2), EM SEGUIDA AS ROTINAS RESTANTES QUE SÃO REFERENCIADAS PELO PROGRAMA OBJETO (GRUPOS 1,3,4) E FINALMENTE AS DEMAIS (GRUPO 5). ESTA ORDEM FOI A ESCOLHIDA, PORQUE FACILITA POSSÍVEIS MUDANÇAS NAS SUBROTINAS SEM IMPLICAR EM MUDANCAS NO COMPILADOR, POIS ESTE NECESSITA CONHECER O ENDEREÇO ABSOLUTO DE ENTRADA DAS SUBROTINAS FUNCIONAIS.

```
*****************************************
*                                       *
* CAPITULO X    * COMPONENTES ADICIONAIS*
*                                       *
*****************************************
```

NESTE CAPÍTULO SERÃO DESCRITOS ALGUNS TÓPICOS QUE VERSAM SOBRE A RELAÇAO ENTRE O SISTEMA E O USUÁRIO. TRATA-SE DE DOIS PROGRAMAS INDEPENDENTES, DESTINADOS A TORNAR O SISTEMA MAIS FLEXÍVEL E PERMITEM AO USUÁRIO AJUSTÁ-LO ÀS SUAS NECESSIDADES.

1. GERAÇÃO DO SISTEMA

HÁ UMA SÉRIE DE PARAMETROS NO SISTEMA, DE DIFÍCIL PREVISÃO A PRIORI, COMO O TAMANHO ÓTIMO DAS DIVERSAS TABELAS, TAIS COMO A DE SÍMBOLOS, DE ARGUMENTOS, DO "STACK" ARITMÉTICO, ETC... PARA SOLUCIONAR ESTE PROBLEMA, CRIAMOS UM "PROGRAMA DE GERAÇÃO", COM O QUAL ESTES PARÂMETROS PODEM SER ALTERADOS À VONTADE, SEM NECESSIDADE DE ALTERAR E REMONTAR O SISTEMA. A IDÉIA DO PROGRAMA CONSISTE EM MODIFICAR, DE ACORDO COM CARTÕES DE CONTROLE, TODOS OS PARAMETROS RELEVANTES DO SISTEMA, QUE POR QUESTÕES DE MODULARIDADE JÁ SE ENCONTRAM NA ÁREA DE COMUNICAÇÕES. OS PARAMETROS DESTE MODO FIXADOS SERÃO USADOS EM TODAS AS UTILIZAÇÕES SUBSEQUENTES, ATÉ QUE NOVA GERAÇÃO SEJA FEITA.

APROVEITANDO A IDÉIA, ÊSTE PROGRAMA DE GERAÇÃO PODE TAMBEM ALTERAR OU FIXAR PARÂMETROS QUE REPRESENTAM OPÇÕES PARA A UTILIZAÇÃO DO SISTEMA, TAIS COMO O LIMITE DE PÁGINAS IMPRESSAS PERMITIDAS DURANTE A EXECUÇÃO DE PROGRAMAS, TEMPO MÁXIMO PERMITIDO DURANTE EXECUÇÕES, IMPRESSÃO OU NÃO DO TEXTO COMPLETO DAS MENSAGENS DE ERROS, IMPRESSÃO OU NÃO DO RELATÓRIO DE "BATCH", IMPRESSÃO OU NÃO DO "SEPARATOR" E O NÚMERO MÁXIMO DE ERROS PERMITIDOS DURANTE A EXECUÇÃO DE UM PROGRAMA.

2. CONTABILIDADE DO SISTEMA

PARA ACOMPANHAR E CONTROLAR A UTILIZAÇÃO DO SISTEMA, CONSIDERAMOS DE GRANDE UTILIDADE A MANUTENÇÃO DE UM ARQUIVO DE CONTABILIDADE. ESTE ARQUIVO, QUE É ATUALIZADO PELO PRÓPRIO SISTEMA, CONTÉM DIVERSAS INFORMAÇÕES SOBRE A SUA UTILIZAÇÃO, TAL COMO O NÚMERO DE JOB'S PROCESSADOS, NÚMERO DE COMPILACOES, NÚMERO DE EXECUÇÕES, NÚMERO DE PÁGINAS IMPRESSAS, NÚMERO DE CARTÕES LIDOS, NÚMERO DE INSTRUÇÕES EXECUTADAS, NÚMERO DE COMPILAÇÕES CORRETAS, NÚMERO DE ERROS DE COMPILAÇÃO, NÚMERO DE ADVERTÊNCIAS DE COMPILAÇÃO, NÚMERO DE EXECUÇÕES CORRETAS, NÚMERO DE ERROS DE EXECUÇÃO E NÚMERO DE PÁGINAS IMPRESSAS EM EXECUÇÕES. PARA CONSULTAR O ARQUIVO, DECIDIMOS FAZER UM PROGRAMA PRÓPRIO, QUE IMPRIME OS CONTEÚDOS DO ARQUIVO, ACOMPANHADO DE ESTATÍSTICAS, COMO O NÚMERO MÉDIO DE CARTÕES POR JOB, NÚMERO MÉDIO DE ERROS POR COMPILAÇÃO, ETC... ÊSTE PROGRAMA PODE TAMBEM, SE DESEJADO, INICIALIZAR O ARQUIVO, GRAVANDO A DATA DE INICIALIZAÇÃO.

CAPITULO XI * TESTE DO SISTEMA

UMA VEZ TENDO A PROGRAMAÇÃO DO SISTEMA COMPLETADA, OS PRIMEIROS TESTES INTEGRADOS PODEM SER REALIZADOS. INICIALMENTE TENTAMOS UTILIZAR PROGRAMAS DE USUÁRIOS PARA REALIZAR A DEPURAÇÃO DO SISTEMA. COMO ESTES PROGRAMAS NAO COBREM TODOS OS CASOS POSSÍVEIS, E PORQUE CADA PROGRAMA NAO UTILIZA APENAS UM RECURSO DO SISTEMA, DE MODO A LOCALIZAR FACILMENTE A REGIÃO DA FALHA, NOTAMOS QUE A DEPURAÇÃO POR ESTE MEIO SERIA MUITO DIFÍCIL E DEMORADA.

PREPARAMOS ENTÃO UM CONJUNTO DE PROGRAMAS DE TESTE, ONDE CADA RECURSO E' EXECUTADO ISOLADAMENTE. ISTO E' CONSEGUIDO DISPONDO-SE A ORDEM DOS PROGRAMAS DE TESTES DE TAL MODO QUE CADA PROGRAMA SOMENTE UTILIZA RECURSOS JA TESTADOS PELOS PROGRAMAS ANTERIORES, ALÉM DO RECURSO ORA SENDO TESTADO. CADA TESTE PROPRIAMENTE DITO CONSISTE EM PROCESSAR CADA PROGRAMA DE TESTE PELO COPPE-FORTRAN E EM SEGUIDA PELO MONITOR. CASO OS RESULTADOS NAO FOREM IDÊNTICOS, DETETAMOS UM ERRO. TEMOS NATURALMENTE DE ESTAR ALERTA CONTRA POSSÍVEIS ERROS DO MONITOR.

SE UM REEXAME DA LISTAGEM DO TRECHO DO SISTEMA QUE CONTEM O ÊRRO NAO O APONTE, UTILIZAMOS PROGRESSIVAMENTE OS SEGUINTES RECURSOS DE DEPURACAO:

1. EM CASO DE PARADA INVÁLIDA DO COMPUTADOR, ANOTAMOS OS REGISTROS DA CPU. ISTO PODE AUXILIAR, PCIS A LISTAGEM EM LINGUAGEM SIMBÓLICA DO SISTEMA E' ABSOLUTA.

2. EXAMINAR "DUMPS" DE MEMÓRIA. EXAMINANDO OS CONTEÚDOS DE CERTAS LOCAÇÕES DA MEMÓRIA, PODE AJUDAR A DESCOBERTA DO ERRO.

3. UTILIZAÇÃO DOS COMANDOS "DUMP COMP" E "DUMP EXEC", COM OS QUAIS OBTEMOS LISTAGENS DAS DIVERSAS TABELAS UTILIZADAS PELO SISTEMA, DURANTE A COMPILAÇÃO E EXECUÇÃO DO PROGRAMA.

4. EMPREGO DO "TRACE": NESTA MODALIDADE PODEMOS ACOMPANHAR INSTRUÇÃO POR INSTRUÇÃO DE MÁQUINA, COM TODOS OS CONTEÚDOS DOS DIVERSOS REGISTROS.

****
* *
* CAPITULO XII *RESULTADOS E CONCLUSÕES*
* *
****

ENTRE OS RESULTADOS POSITIVOS, TEMOS NATURALMENTE O DE QUE O SISTEMA ENCONTRA-SE EM FUNCIONAMENTO COM SUCESSO. NO SEU PRIMEIRO MES DE UTILIZAÇÃO (MAIO DE 1972), PROCESSOU MAIS DE 2400 JOBS, COM UMA VELOCIDADE APROXIMADAMENTE 3 VEZES MAIOR DO QUE O MONITOR, NA CONFIGURAÇÃO COM 32K DE MEMÓRIA, IMPRESSORA 1403 E LEITORA 2501. ESTE VALOR FOI OBTIDO COM UM "BATCH" DE TESTES.

MEDIDAS EFETUADAS MOSTRARAM QUE DE ACÔRDO COM A CONFIGURAÇÃO DO 1130 E OS PARAMETROS DE GERAÇÃO ESCOLHIDOS, A RELAÇÃO DE VELOCIDADES ENTRE O COPPE-FORTRAN E O MONITOR VARIA DESDE 1,4 A 3,7 (MEDIDO COM UM "BATCH" DE TESTES). O AUMENTO DE "THROUGHPUT" FOI TAMBEM CONSIDERÁVEL, VISTO QUE EM MÉDIA, O NÚMERO DE PROGRAMAS PROCESSADOS POR DIA SUBIU DE 250 PARA 350, EMPREGANDO-SE NATURALMENTE O COPPE-FORTRAN APENAS PARA OS PROGRAMAS SIMPLES (MEDIDO EM USO REAL).

CONCLUÍMOS TAMBÉM QUE A TÉCNICA RESIDENTE E' TAMBEM VÁLIDA PARA COMPUTADORES PEQUENOS DE UNIVERSIDADES. PORTANTO, A IDÉIA DEVE SER UTILIZADA POR OUTRAS UNIVERSIDADES QUE TENHAM PROBLEMAS ANALOGOS AOS AQUI DESCRITOS.

ENTRE OS RESULTADOS POSITIVOS TEMOS TAMBEM DE QUE O SISTEMA NAO SERÁ SOMENTE ÚTIL PARA O NOSSO CENTRO DE COMPUTAÇÃO, MAS PODERÁ TAMBÉM SER UTILIZADO POR CERCA DE 20 OUTRAS UNIVERSIDADES QUE TAMBEM POSSUEM O 1130.

E' TAMBEM UM RESULTADO POSITIVO O APRENDIZADO NO CAMPO DE COMPILADORES E O AUMENTO DE CONHECIMENTO EM DIVERSOS CAMPOS LATERAIS, COMO POR EXEMPLO, A UTILIZAÇÃO DO MACRO-ASSEMBLER.

EXAMINANDO O SISTEMA DEPOIS DE COMPLETADO, PODEMOS VER QUE EMBORA EXISTAM SEMELHANÇAS ESTRUTURAIS ENTRE O "COPPE-FORTRAN" E O "WATFOR", SOMENTE O ALGORITMO DE DETEÇÃO

DE DESVIOS INVÁLIDOS PARA O INTERIOR DE MALHAS DE COMANDOS "DO" FOI BASEADO DIRETAMENTE NO "WATFOR".

UMA CONCLUSÃO INTERESSANTE É' A DE QUE O AUMENTO DE EFICIÊNCIA QUE O SISTEMA PROVÊ É' RESULTANTE TANTO DA COMPILAÇÃO MAIS VELOZ, COMO DA AUSÊNCIA DE "LINK-EDIÇÃO" E TRANSIÇÃO DE "JOB" PARA "JCB" MAIS RÁPIDA.

É' IMPORTANTE TAMBEM NOTAR QUE A DOCUMENTAÇÃO DO MÉTODO DE ELABORAÇÃO DO SISTEMA PERMITIRÁ UM USO DIDÁTICO, EM VIRTUDE DA FALTA DE LITERATURA EM PORTUGUÊS SÔBRE O ASSUNTO.

O SISTEMA PODERÁ SER UTILIZADO COMO UM PRÉ-COMPILADOR PARA PROGRAMAS FORTRAN EM GERAL, VISTO QUE APESAR DE NAO PODER EXECUTAR TODOS OS PROGRAMAS, AUXILIARÁ MUITO PARA A DETEÇÃO DE ERROS DE COMPILAÇÃO, EM VIRTUDE DA SUA VELOCIDADE E QUALIDADE DOS DIAGNOSTICOS QUE PROVE.

UM RESULTADO NEGATIVO QUE FOI CONSTATADO É' O DE PARA A ELABORAÇÃO DO SISTEMA FOI NECESSÁRIO UM TEMPO MUITO MAIS LONGO DO QUE O PREVISTO (FORAM NECESSÁRIOS 2 ANOS E 2 MESES). ISTO OCORREU PRINCIPALMENTE PELO TRABALHO ENORME DE PROGRAMAÇÃO ENVOLVIDO. PODEMOS DAI TIRAR 2 CONCLUSÕES: O DESENVOLVIMENTO DE UM SISTEMA DESTE TIPO É' COMPLEXO DEMAIS PARA UMA TESE DE MESTRADO, E QUE PORTANTO O DESENVOLVIMENTO DEVE SER FEITO POR UMA EQUIPE, E NAO INDIVIDUALMENTE.

UMA IDÉIA QUE TALVEZ SEJA INTERESSANTE PARA A CONTINUAÇÃO DE PESQUISA NO CAMPO, É' A INTEGRAÇÃO DO SISTEMA "BASIC - IBM-1130" COM O COPPE-FORTRAN COM A UTILIZAÇÃO DE UM SÓ SUPERVISOR, DE MODO A PROCESSAR "BATCH" DE PROGRAMAS BASIC E FORTRAN COM GRANDE EFICIENCIA. DESTE MODO ESTARÍAMOS UNINDO EM UM SÓ SISTEMA DUAS LINGUAGENS MUITO UTILIZADAS NO ENSINO DE COMPUTAÇÃO.

DURANTE O INÍCIO DA ELABORAÇÃO DO SISTEMA, PENSAMOS EM POSTERIORMENTE INCORPORAR UM SISTEMA DE "SPOOL" DE ENTRADA/SAIDA. VERIFICAMOS, NO ENTANTO, QUE O SISTEMA PERMANECE PRATICAMENTE DURANTE TODO O TEMPO DE UTILIZAÇÃO, COM AS UNIDADES DE ENTRADA/SAIDA EM FUNCIONAMENTO. ISTO SIGNIFICA QUE UM SISTEMA DE "SPOOL" IRÁ TRAZER POUCO OU ATÉ NENHUM BENEFÍCIO AO SISTEMA.

O SISTEMA, TAL COMO FOI ELABORADO, PODE SER UTILIZADO COM 1130'S COM 32K E COM 16K. A PERDA DE EFICIÊNCIA EM 16K EM VIRTUDE DE OVERLAYS É' PEQUENA, E AINDA É' BEM VANTAJOSA A SUA UTILIZAÇÃO, MESMO COM 16K.

REFERÊNCIAS BIBLIOGRAFICAS

1. DAVIDSON, C. H., KOENIG, E. C.: "COMPUTERS: INTRODUCTION TO COMPUTERS AND APPLIED COMPUTING CONCEPTS". JOHN WILEY, 1967, PG. 20.

2. CONWAY, R. W., MAXWELL, W. L.: "CORC - THE CORNELL COMPUTING LANGUAGE". COMMUNICATIONS OF THE ACM, JUNHO DE 1963, 317-321.

3. ROSEN, S. ET AL.: "PUFFT - THE PURDUE UNIVERSITY FAST FORTRAN TRANSLATOR". COMMUNICATIONS OF THE ACM, NOVEMBRO DE 1965, 661-666.

4. SHANTZ, P. W. ET AL.: "WATFOR - THE UNIVERSITY OF WATERLOO FORTRAN IV COMPILER". COMMUNICATIONS OF THE ACM, JANEIRO DE 1967, 41-44.

5. CRESS, P. H., DIRKSEN, P. H. ET. AL.: "DESCRIPTION OF /360 WATFOR, A FORTRAN IV COMPILER". CSTR-1000, APPLIED ANALYSIS AND COMPUTER SCIENCE DEPARTMENT, UNIVERSITY OF WATERLOO, ABRIL DE 1968.

6. IBM CORPORATION: "IBM-1130 FUNCTIONAL CHARACTERISTICS". SRL GA26-5881-5.

7. IBM CORPORATION: "IBM-1130 DISK MONITOR SYSTEM, VERSION 2: PROGRAMMING AND OPERATOR'S GUIDE". SRL C26-3717-3.

8. IBM CORPORATION: "IBM-1130 DISK MONITOR PROGRAMMING SYSTEM, VERSION 2: PROGRAM LOGIC MANUAL". SRL Y26-3714-1.

9. IBM CORPORATION: "IBM-1130/1800 BASIC FORTRAN IV LANGUAGE". SRL C26-3715-3.

10. IBM CORPORATION: "IBM-1130 SUBROUTINE LIBRARY". SRL C26-5929-4.

11. IBM CORPORATION: "IBM-1130 ASSEMBLER LANGUAGE". SRL C26-5927-4.

12. IBM CORPORATION: "IBM-1130 MACRO ASSEMBLER LANGUAGE". SRL GC26-3733-0.

13. GRIES, D.: "COMPILER CONSTRUCTION FOR DIGITAL COMPUTERS". JOHN WILEY, 1971.

14. MORRIS, R.: "SCATTER STORAGE TECHNIQUES". COMMUNICATIONS OF THE ACM, JANEIRO DE 1968, 38-44.

15. MAURER, W. D.: "AN IMPROVED HASH CODE FOR SCATTER STORAGE". COMMUNICATIONS OF THE ACM, JANEIRO DE 1968, 35-38.

16. GALLER, B. A., FISHER, M. J.: "AN IMPROVED EQUIVALENCE ALGORITHM". COMMUNICATIONS OF THE ACM, MAIO DE 1964, 301-303.

17. HOPGOOD, F. R. A.: "COMPILING TECHNIQUES". MAC DONALD, LONDRES, 1970.