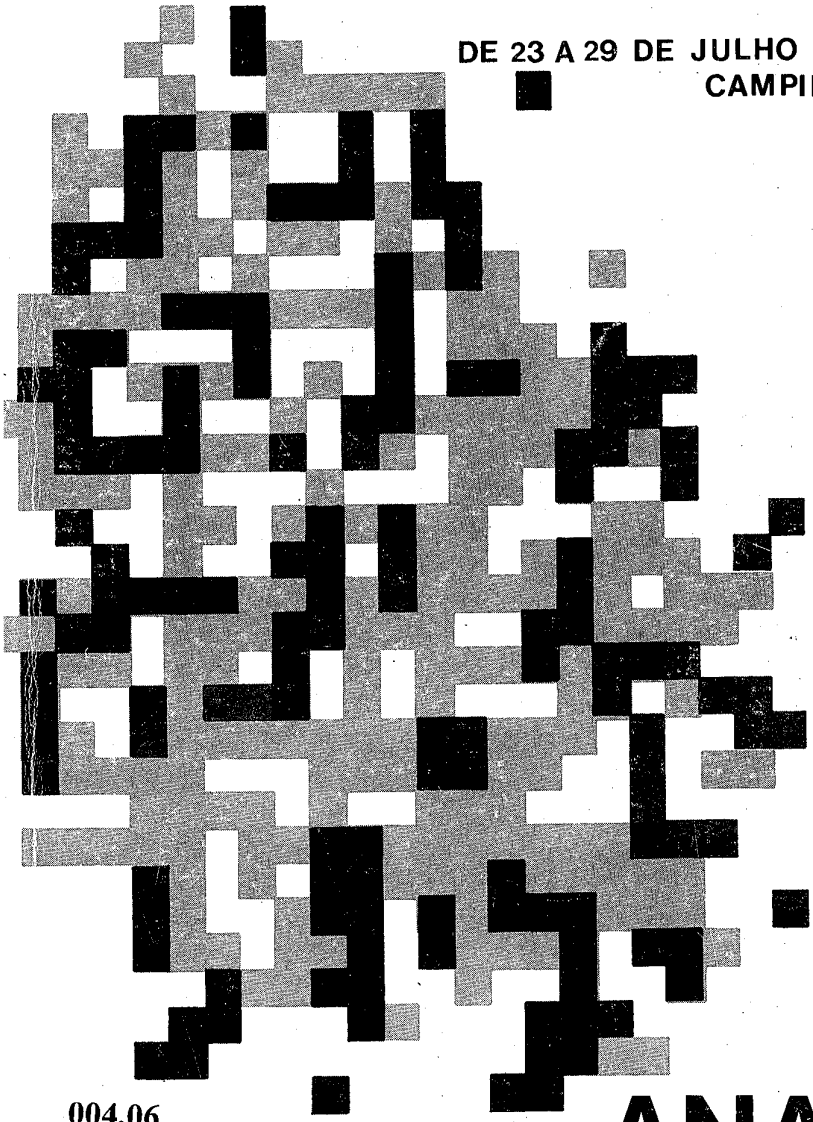
III CONGRESSO

DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

DE 23 A 29 DE JULHO DE 1983
CAMPINAS - SP



ANAIS

VOL. I

III CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO
CAMPINAS 23 a 29 de julho de 1983

ANAIS
VOLUME I

TRABALHOS APRESENTADOS
X SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE

EDITORES: N. MEISEL, L.J. BRAGA-FILHO

PROMOÇÃO: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO - SBC
UNIVERSIDADE ESTADUAL DE CAMPINAS - UNICAMP

PATROCÍNIO: CAPES, CNPq, DIGIBRÁS, FINEP, SEI

CO-PATROCINADORES: CPqD/TELEBRAS, PUC-Campinas, UFV
PREFEITURA MUNICIPAL DE CAMPINAS

UMA FERRAMENTA DE APOIO AO ESTUDO DE MICROPROGRAMAÇÃO

C. S. Menezes+ A.v.Staa*

SUMÁRIO

No presente trabalho apresenta-se um sistema de Simulação de Micromáquinas . Tal sistema permite a especificação e realização de Micromáquinas em ambientes de Soft - ware. Dispondo de uma linguagem de descrição possibilitamos seu uso em um número variado de Micromáquinas. Uma base operacional possibilita a "realização" de micromáquinas descritas, submetendo microprogramas. O exercício de microprogramas pode ser acompanhado numa sessão de simulação através de uma linguagem de controle.

ABSTRACT

A micro-machine simulator is described. This system allows the specification and simulation of computers at the bit-slice level. By means of a machine description language a wide variety of micro-machines may be simulated. The system is used to simulate the behaviour of micro-programs to be executed on described machines. The evolution of the micro-program may be monitored by means of a control language.

+ Licenciado em Matemática(Univ. AM-1976), Mestre em Ciência em Informática(PUC/RJ-1983), em Doutoramento(PUC/RJ), Prof. Assistente (Univ. do Amazonas); especificações, Engenharia de Software.

* Engenheiro Mecânico(PUC/RJ-1965), Mestre em Informática(PUC/RJ-1969), Doutor em Ciência da Computação(Univ. Waterloo-1974); Prof. Associado(PUC/RJ); Especificações,Linguagens de Programação, Engenharia de Software.

(+,*) ENDEREÇO - Pontifícia Universidade Católica do Rio de Janeiro - Depto. de Informática, Rua Marques de São Vicente, 225- Gávea - CEP - 22453 - Rio de Janeiro - Brasil.

1. Introdução

Sistemas de computação são realizados através de várias camadas, chamadas máquinas virtuais. Por exemplo linguagem de alto nível, sistemas operacionais, assembler , máquina nua, micromáquina formam uma hierarquia de máquinas [M80].

O nível comumente apresentado como hardware é a máquina nua. Esta pode ser implementada em micromáquinas através de circuitos combinacionais ou microprogramação[M80].

Quanto à realização de micromáquinas, estas tanto podem ser construídas sob medida ou utilizando-se de famílias de blocos pré-construídos (bit-slices) [M80].

Em que pesem as facilidades introduzidas pelo uso de microprogramação e bit-slices, o projeto, avaliação e estudo de arquiteturas carecem de ferramentas de apoio.

O presente trabalho apresenta as características, a especificação e o projeto de um Simulador de Micromáquinas [M82], desenvolvido no Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro (PUC/RJ).

Tal sistema tem como função principal permitir a especificação e realização de micromáquinas em ambiente de software e tem como objetivo contribuir para:

- auxiliar na verificação e validação de novas arquiteturas utilizando um ambiente ameno ao uso, longe de circuitos, fios, placas e outros apetrechos ;
- criação e teste de microprogramas para máquinas existentes, sem que essa precise ser desalocada de suas tarefas rotineiras;
- facilitar o aprendizado de organização de computadores, fundamentos de microprogramação e algorítmos básicos de um computador.

O sistema desenvolvido forma uma base operacional onde é possível a simulação de um número virtualmente ilimitado de micromáquinas. Esta flexibilidade é obtida através de um processador de Descrição que recebe a descrição de uma micromáquina gerando tabelas a serem usadas pelo simulador. O programa simulador é um esqueleto operacional que deve ser acrescido de um código ativador de dispositivos (também gerado pelo Processador de descrição) e das rotinas específicas para realização de funções eletrônicas. As rotinas específicas tanto podem fazer parte de uma biblioteca quanto serem construídas pelo usuário.

O sistema desenvolvido não prevê uma linguagem de alto nível para microprogramação nem um gravador de Micromemórias como os apresentados em [Y82, 082], sendo no entanto mais geral visto que permite o uso na simulação de diferentes micromáquinas

2. Características do Sistema

Para possibilitar que especificações e microprogramas possam ser verificados

sem grandes atropelos, e com alto grau de confiabilidade, dotamos o sistema das seguintes características:

a) Facilidade para descrever micromáquinas - o sistema é provido de uma linguagem para especificar micromáquinas, flexível o bastante para permitir a sua utilização no projeto de diferentes micromáquinas.

b) Facilidade para incorporar novas funções eletrônicas-o sistema é provido de mecanismo que possibilita ao usuário escrever e utilizar rotinas que implementem novas funções eletrônicas, tais como um multiplexador, um circuito decodificador ou um determinado bit-slice.

c) Facilidade para controlar o funcionamento da " máquina " - o sistema é dotado de uma linguagem que permite ao usuário controlar a execução de um micro programa. Tal linguagem é provida de mecanismo para carregar microprogramas, indicar pontos de parada durante a execução de um microprograma (break-points), mostrar o conteúdo de registradores e memórias, atribuir valor a registradores e memórias, retomar a execução a partir de um dado endereço de micromemória e monitorar recursos da máquina.

d) Outras - o simulador leva em conta as características físicas dos componentes da micromáquina, para que tenha utilização como instrumento de apoio à verificação do projeto. Ao final de cada sessão é exibido um mapa de utilização de recursos e tipos de microinstrução.

3. Especificação Funcional

Como já enfatizado nas características do sistema o uso do Sistema de Simulação vai muito além do exercício de Microprogramas. Na verdade, Microprogramas necessitam de teste de correção, antes que outras propriedades possam ser avaliadas.

Além do teste de Microprograma [M80], na verdade um pouco antes, devemos verificar a micromáquina especificada. A verificação de uma micromáquina, consiste básicamente em testar as ligações entre componentes e verificar a correção das funções que realizam dispositivos. Sendo que estas últimas podem ser verificadas sob controle do simulador.

Para torná-lo flexível, dotamos o sistema de um processador de descrição, que permite seu uso na simulação de qualquer micromáquina.

Na figura 1 apresenta-se o fluxo de dados geral do problema.

3.1 - Abstração de Micromáquinas

O modelo adotado para descrever micromáquinas, tem como base que uma micromáquina é um conjunto de dispositivos funções de via, funções eletrônicas e unidades de memória que operam sobre determinados recursos - registradores, partes de registradores, ou vetor de registradores funcionando em determinados subciclos e controlados por código de operação oriundo de um determinado recurso (ver fig. 2).

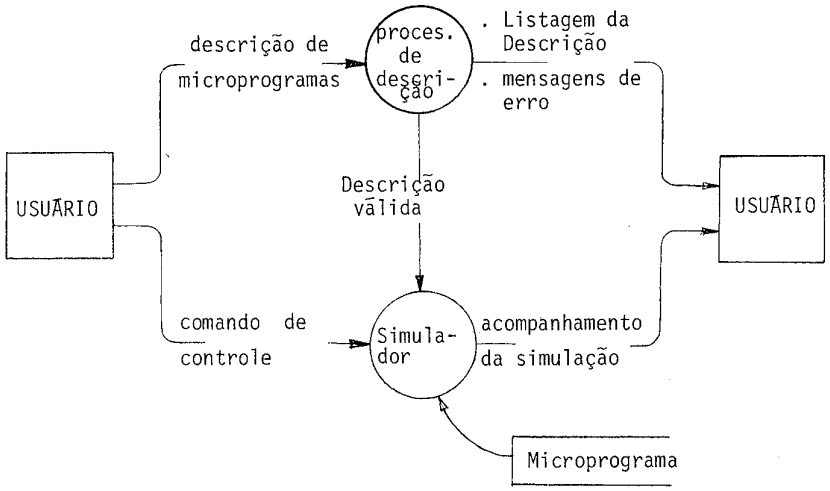


FIG. 1 - FLUXO DE DADOS DO SISTEMA

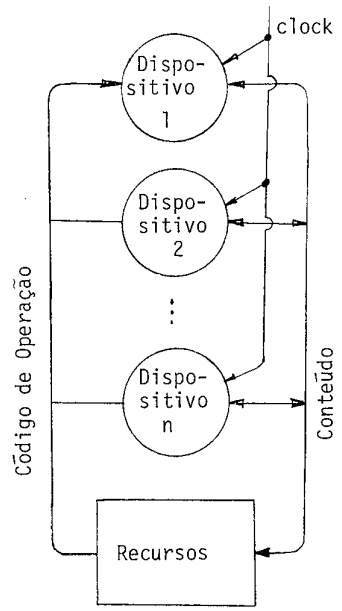


FIG. 2 - ABSTRAÇÃO DE UMA MICROMÁQUINA

Funções de via, são aquelas que ligam dois ou mais registradores, cuja finalidade é controlar o fluxo de dado entre eles ("gates", "shifters",...).

Funções eletrônicas usam o conteúdo de registradores, para produzir um novo dado que será transferido para um ou mais registradores. Podem ser simples como um contador ou complexas como um bit-slice que implementa uma unidade de processamento. Uma função pode inclusive ser dotada de memória interna.

Unidades de memória são vistas, em nosso modelo, como funções eletrônicas com memória interna, que recebem um endereço e recuperam ou armazenam um valor de/em um outro registrador, controladas por um código de operação.

As definições acima permitem-nos uma abstração de qualquer função da máquina como um dispositivo, que usa recursos e possui ou não memória interna.

Um registrador físico pode ser dividido logicamente em partes (registrador lógico). E tanto um como outro serão tratados por registrador.

Algumas funções usam registradores bem determinados na entrada e na saída, outras usam o conteúdo proveniente de um registrador selecionado por uma outra função , e há ainda outras cujo resultado da ativação será armazenado em um ou mais registradores condicionados à seleção de uma outra função usa como entrada/saída uma via de dados que a liga com uma outra função.

Apenas nos casos de indefinição acima expostos, sente-se a necessidade de ver uma via como um recurso. A solução adotada consiste em criar pseudo-registradores que servirão de interface entre os dispositivos envolvidos, generalizando desta forma o modelo.

Um dispositivo entra em funcionamento sempre que acionado pelo relógio e os recursos necessários estejam disponíveis. Os recursos podem ser dados operacionais ou de controle (código de operação), sendo que os de controle são em geral oriundos de uma microinstrução.

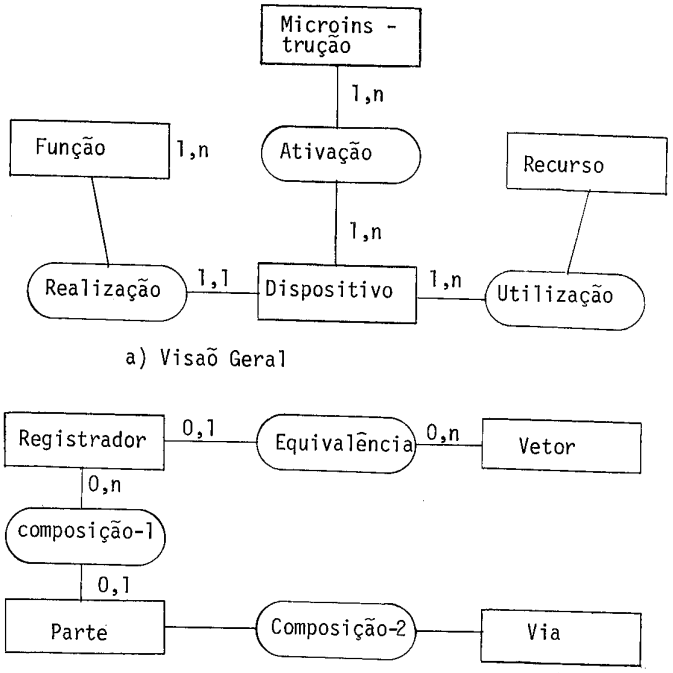
Para controlar todos os dispositivos, uma microinstrução necessitaria prover um campo para cada um deles. Por questões técnicas e econômicas isto não é conveniente e o que se tem é um conjunto de microinstruções, onde cada tipo de microinstrução controla grupos de dispositivos. Cada tipo de microinstrução é reconhecido por uma configuração de bits.

A figura 3 a seguir sintetiza o modelo de dados [I82, S80] do sistema.

3.2 - Linguagem de Especificação de Micromáquinas

Na linguagem proposta uma máquina é descrita através de seus recursos, dispositivos e tipos de microinstrução.

A seguir apresentamos um resumo da linguagem através de exemplos. Maiores detalhes podem ser obtidos em [M83]



a) Visaõ Geral

b) Detalhamento da Entidade-Tipo Recurso

FIG. 3 - MODELAGEM DE DADOS DO SISTEMA

3.2.1 Descrição de Recursos

a) Registradores e suas partes

```
$REG  ALFA,16
  $P  A,11,15
  $P  B,1,10
  $P  B1,1,5
  $P  B2,6,10
  $P  C,8,13
```

b) Vetor de Registradores

```
$VET  MEMÓRIA,32,8192
```

descreve um vetor de nome memória, com 8192 elementos e de largura 32.

c) Via de Dados

```
$VIA  TEMP1,16
```

descreve uma via de nome TEMP1 com largura 16.

d) Equivalencia de Registradores

```
$REG   PC,16
$VET   VETA,16,15
$EQV   PC = VETA(7)
```

descreve um registrador PC, um vetor VETA e a equivalência entre PC e o sétimo elemento de VETA.

3.2.2 Dispositivos

A descrição de um dispositivo, envolve a associação deste com uma rotina do usuário que o realizará, a relação dos recursos utilizados pelo dispositivo, e informações sobre tempo de ativação. Exemplo:

```
$DISP  MUX1:  MUTEX(SVIA1,A,B,VIA1),1,2,1
```

Descreve um multiplexador(MUX1) implementado pela rotina MUTEX, o qual coloca em uma via de dados (VIA1) o conteúdo de um registrador A ou B, de acordo com o código de seleção oriundo do bit 10 do registrador de microinstruções(MICROIR). O dispositivo entra em funcionamento no primeiro subciclo, sua ordem de ativação é 2 e dispende um subciclo.

3.2.3 Microinstruções

Cada micromáquina pode possuir um ou mais tipos de microinstrução. Cada tipo de microinstrução é responsável pela ativação de um grupo de dispositivos. A cada tipo de microinstrução está associado um conjunto de bits do MICROIR que as identifica. Exemplo

```
$MINST PROCESS,'1XO11XO', GATE1,ALU,GATE2
```

Especifica a microinstrução PROCESS, reconhecida quando os bits 6, 3 e 2 do MICROIR estiverem ligados e os bits 4 e 0 desligados. A microinstrução PROCESS é responsável pela ativação dos dispositivos GATE1,ALU e GATE2.

3.3 Linguagem de Controle da Execução

Para controlar o funcioanmento da "máquina", e provido uma linguagem de comando, que permite ao usuário um completo domínio sobre a execução de um microprograma. A seguir apresentamos um resumo, ao leitor interessado sugerimos [M83].

A versão inicial da linguagem permitirá:

a) seleção de microprogramas para carga,

```
$CARGA  15 - carrega um microprograma a partir do endereço 15 da Micromemória
```

b) seleção do ponto de partida para execução de um microprograma, com o controle de parada.

```
$EXECUTE 15   $REF micmem=10
```

inicia a execução de um microprograma a partir da posição 15 da microme-

mória e é interrompida quando o número de referências ao recurso MICMEM atingir 10 vezes.

c) inspeção e modificação do conteúdo de um recurso.

i) $MOSTRE alfa,micmem(115..200)

exibe o conteúdo do registrador ALFA e dos elementos de endereço 115 a 200 do vetor de registradores MICMEM.

ii) $ALTERE alfa=beta

atribui ao recurso ALFA o conteúdo do registrador BETA.

d) monitoramento de recursos

$ARASTRO mem01,beta,stack

ativa o rastreamento sobre os recursos MEM01, BETA e STACK.

4. Projeto Lógico

O sistema foi dividido em 2 subsistemas, os quais se comunicam através de 2 estruturas de dados. O primeiro é o processador de Descrição que analisa a descrição de uma Micromáquina. Para descrições corretas é produzida uma tabela descritora de micromáquinas e um código (em linguagem fonte) ativador de dispositivos.

Como para cada máquina será gerado um ativador de dispositivos, há então a necessidade de agregar este código ao simulador.

O subsistema seguinte é o simulador.

Este sob controle de uma linguagem de comandos executará microprogramas na máquina descrita.

4.1 - Realização de Dispositivos

As funções eletrônicas de uma micromáquina, aqui chamadas dispositivos, podem ser as mais variadas possíveis, sendo pois impossível embuti-las todas no simulador.

Duas soluções principais foram vislumbradas. A primeira delas consistia no desenvolvimento de uma linguagem de descrição mais poderosa, de tal forma que pudessemos descrever as operações que realizam um dispositivo. A outra, bastante mais simples, consistia em agregar ao simulador trechos de programas escritos em uma linguagem de programação.

A primeira foi descartada dada a exiguidade de tempo destinada ao projeto.

Nosso sistema usa portanto a segunda opção. Para facilitar e proteger a manipulação dos recursos da máquina, são providas rotinas para acesso e modificação de seus conteúdos.

4.2 - Base de Dados

A base de dados do sistema é formada pela representação das entidades e relacionamentos do sistema, além das palavras chaves das duas linguagens.

As entidades são mantidas em lsitas lineares, com a cabeça da lista indicada pe

la palavra chave correspondente da linguagem de descrição. Por exemplo $REG indica a cabeça da lista de registradores.

Os relacionamentos-tipo composição-1, composição-2, ativação e utilização são mantidos em listas lineares, uma para cada instância do relacionamento. O relacionamento 'realização' é mantido com um campo de Informação dos dispositivos. Equivalencia entre registradores e vetores é traduzida pela igualdade de conteúdo.

Para dar maior flexibilidade às dimensões dos recursos e ao mesmo tempo obter economia no armazenamento, criamos uma memória do simulador onde são armazenados os conteúdos dos recursos.

Associada a base de dados acima descrita, existe um conjunto de operações para manipular as estruturas. O modelo lógico da Base de Dados mais o conjunto de Operações formam o Tipo Abstrato de Dados, que chamaremos TABELA DE SIMBOLOS:

As Operações definidas são: Instala objeto, encadeia objeto, encadeia dispositivos, encadeia recursos, encadeia partes, obtém valor de atributos, atribui valor a atributo, obtem conteúdo de um recurso e atribui valor ao conteúdo de um recurso.

O conteúdo de cada recurso é uma sequência de bits. Para facilitar sua manipulação, definiu-se o tipo abstrato REGIST com as seguintes operações: PREENCHER COM ZEROS , PREENCHER COM UNS, MOVER DADOS DE UM REGIST PARA OUTRO, converter o valor de um REGIST para decimal.

4.3 - Processador de Descrições

Seguindo os algoritmos e a metodologia apresentada em [S81], usamos a estratégia de Análise Sintática dirigida pelo grafo sintático da linguagem. A simplicidade da linguagem possibilitou a construção de um grafo composto apenas por simbolos terminais o que implicou numa simplificação do algoritmo de reconhecimento.

O tratamento de erro usa apenas a estratégia de busca do delimitador, o qual é reconhecido pela presença de uma palavra chave

Como produto final deste processador, temos: A tabela descritora da micromáquinas especificada no formato adequado ao simulador e o código fonte ativador de dispositivo.

O código ativador de dispositivos é uma subrotina que dado o conteúdo do Registrador de Microinstruções ele determina e ativa o conjunto de funções que realizam os dispositivos ativados por aquele tipo de microinstrução. As funções são ativadas sequencialmente, por ordem de subciclo de ativação.

4.4 - Simulador

O funcionamento do Simulador é calcado na interpretação dos comandos da linguagem de controle. A estratégia adotada para reconhecimento desta linguagem é a mesma utilizada no processador de descrições. A cada vez que é encontrado um nó (grafo sintático) que identifica uma ação, a rotina semântica associada ao nó ativa o processo correspondente.

5. Projeto Físico

Nos itens seguintes apresentamos as principais características adotadas na implementação do sistema. Em alguns casos, o principal fator que contribui para a escolha de uma alternativa foi a exiguidade de tempo destinado ao projeto.

5.1 - Linguagem de Programação

Um dos principais critérios de qualidade que norteiam a implementação de um software desta classe, é sem dúvida alguma a portabilidade. É obvio que esta é, em geral, conflitante com eficiência, não menos importante que a primeira.

É um consenso geral, entretanto, que a eficiência de um sistema pode ser ajustada após uma primeira implementação (protipo), principalmente quando esta está estritamente relacionada com a linguagem de programação.

A linguagem PASCAL foi portanto adotada, levando em conta que a) A linguagem é de alto nível o que facilita a implementação; b) O uso desta linguagem vem crescendo ultimamente; c) A linguagem é utilizada em uma grande quantidade de microcomputadores (ambiente ideal para a ferramenta)

5.2 - Estruturas de Dados

Os objetos do sistema são encapsulados em um array de variant records. Cada objeto é identificado por um nome, sobre o qual pode ser feita a recuperação. O acesso a esta tabela é feito por cálculo de endereço (HASH) e o tratamento de colisão por endereçamento aberto (Open Addressing).

As varias listas que implementam relacionamentos, são implementadas por encadeamento no processador de descrição enquanto no simulador a implementação é mista. No simulador, a lista de "Ativação" é transformada em sequência de ativação de funções, a lista de "Utilização" em parâmentro das funções, enquanto as demais permanecem por encadeamento.

A memória do simulador é um array de variáveis booleanas. Para cada recurso é reservado um número de elementos da memória, compatível com sua largura e elementos (só para vetores). As partes de um registrador ou via, ocupam parte do espaço físico destes. Uma equivalência é representada através da associação de endereço. No registro de cada recurso há um campo onde é armazenado o endereço na memória correspondente ao bit de mais alta ordem do recurso.

Os grafos sintáticos são implementados em um array, onde cada nó possui os campos: simbolo, alternativa, sucessor e rotina semantica. Os campos, alternativa e sucessor são indices no array onde se encontram armazenados os simbolos que são alternativa e sucessor respectivamente do simbolo final é identificado por ter o sucessor igual a zero, enquanto o simbolo inicial é guardado no primeiro elemento de array. O campo rotina semântica guarda um número que identifica um conjunto de comandos a serem selecionados atraves de um 'case'.

5.3 - Rotinas Semânticas

Para cada analisador (sintático ou léxico) há rotinas semânticas associadas a cada simbolo reconhecido. Para cada analisador as rotinas são agrupadas em uma única 'Procedure'. A seleção de uma rotina semântica é feita através de um parâmetro (ação) que serve de argumento a um seletor 'case'. A cada valor do parâmetro esta associado um conjunto de comandos que realizam a rotina semântica correspondente.

6. Conclusões

Neste trabalho apresentamos a especificação e construção de um sistema de simulação de micromáquinas. Muito embora seja este o resultado mais tangível, este não é o único objetivo. Apesar do sistema produzido constituir-se numa ferramenta útil, nosso objetivo maior era a utilização de conceitos e metodologias de Engenharia de Software na construção de um sistema não trivial.

Para avaliar a validade do ferramental utilizado faz-se necessário observar dois aspectos. Em primeiro lugar a complexidade do sistema. Apesar de não se constituir em um sistema altamente complexo, o sistema produzido possui algumas características que o tornam um sistema não trivial. Destas, destaca-se a necessidade da criação de um modelo para o objeto da simulação, seguida do projeto e implementação das linguagens de descrição e controle. Em segundo lugar, o tempo decorrido desde a concepção até a implementação do sistema. A concepção, especificação e projeto foram feitas ao longo de cinco meses em dedicação parcial enquanto que a implementação (codificação e testes) foi realizada em um mes.

Como metodologia, destacamos a adoção de um ciclo de vida bem definido (concepção, especificação funcional, projeto físico, projeto lógico e implementação)[S79 , S82] e modularização [Y79]. A primeira permitiu a maturação e validação gradativa do produto a ser produzido, enquanto a segunda permitiu a quebra do sistema em módulos funcionais, obviamente mais faceis de tratar.

Como conceitos, podemos destacar o meta-modelo de entidades, relacionamentos e atributos [I80 , S80] que ajudou a capturar o modelo de simulação a partir do qual tornou-se fácil derivar a linguagem de descrição. O uso de fluxo de dados[O79] facilitou a análise do sistema como um todo, além de contribuir para o projeto modular do simulador e à derivação da linguagem de controle. Para armazenar dados, utilizou-se desde o início do projeto o conceito de tipo abstrato de dados [P81], o que nos permitiu uma implementação e uso correto e preciso dos objetos do sistema.

O sistema implementado pode ser avaliado através de um confronto direto entre a especificação de requisitos e o exemplo rodado no sistema. Apesar disso, é bom considerar dois aspectos. A flexibilidade e amenidade ao uso foram perfeitamente cobertas pela introdução das duas linguagens. Quanto às características funcionais de uma sessão de simulação, também foram atingidas, muito embora a validação dos instrumentos de auxilio só possa ser totalmente realizada após um uso mais intenso.

Para finalizar nossos comentários sobre o sistema produzido, gostariamos de dei-

xar registrado que há necessidade de se agregar outras ferramentas à ele. Um subsistema para preparação de microprogramas e um gravador de memórias ROM, aumentarão sensivelmente a utilizabilidade do sistema.

7. REFERENCIAS

[D79] - De Marco, T. - Structured Analysis and System Specification, Prentice-Hall, USA, 1979

[I82] - ISO-Concepts and Terminology for the conceptual Schema and Information base, relatório TC97/SC5/WG3, editado por J.J. van Griethaysen, mar/82

[M79] - Mezzalona, M., Prinotta, P. - Design and Implementation of a Flexible and Interactive Microprogram Simulation, em SIGMCRO-newletter, vol 10 , nº 4, dez/79

[M80] - Myers, G.J. - Digital System Design with Bit-slice Logic, USA, J.Willey , 1980

[P81] - Pessoa, F.E.P. - Programacão-com tipos de Dados Abstratos, Dissertação de Mestrado, deptº Informática, PUC/RJ, 1982

[S79] - Staa, A.v. - Desenvolvimento Estruturado de Sistemas Automatizados, Monografias em Ciência da Computação, PUC/RJ, 1979

[S80] - Santos, C.S. - Caracterização Sistematica de Restrições de Integridade em Banco de Dados, Tese de Doutorado, Deptº de Informática, PUC/RJ, 1980

[S81] - Setzer, V.W., Melo, I.S.H. - A Construção de um Compilador, segunda Escola de Computação, UNICAMP/SP, 1981

[M82] - Menezes, C.S., Staa, A.v. - Um Simulador de Micromaquinas, Anais do 2º Simpósio de Software Básico para Microcomputadores, SBC/USP, São Paulo , dez/82

[M83] - Menezes, C.S. - Um Simulador de Micromaquinas: Especificação, Projeto e Implementação, Dissertação de Mestrado, Deptº Informática, PUC/RJ, mar/83.

[Y82] - Yamaguti, W. - Desenvolvimento de Sistemas Microprogramados - Geração de Microcódigos, Anais do IX SEMISH, 1982.

[O82] - Oliveira, F.J. de; Cox, P.H., Soares, L.F.G. - Sistema de Auxílio ao Desenvolvimento de Sistemas Microprogramados, Anais do IX SEMISH, 1982.

[S82] - Staa, A.v. - Engenharia de Programas, 3ª Escola de Computação, PUC/RJ, 1982

[Y79] - Yourdon, E., Constantine, L.L. - Structured Design, Prentice-Hall, USA,1979