

UNESP
Faculdade de Engenharia de Guaratinguetá

Guaratinguetá
1997



Galdenoro Botura Jr

CARO LEITOR NÃO
RISQUE O LIVRO

Doutor em Engenharia Elétrica, UNICAMP 1991

Mestre em Engenharia Elétrica, UNICAMP 1985

Engenheiro Eletricista, INATEL 1980

Proposta de um Dispositivo ASIP de um Micro- controlador RISC de 32 Bits

Trabalho apresentado como parte dos requisitos exigidos para concurso de Livre-Docência, na disciplina de Eletrônica Digital do Departamento de Engenharia Elétrica da Faculdade de Engenharia do Campus de Guaratinguetá - UNESP.

333

Universidade Estadual Paulista
Faculdade de Engenharia
Campus de Guaratinguetá
Departamento de Engenharia Elétrica

333

Guaratinguetá, Março de 1997

FACULDADE DE ENGENHARIA
DE GUARATINGUETÁ
BIBLIOTECA

T621.3.049.77 (043)
B749p

- II -



À Margarete e a Erika

*Pelo apoio, compreensão e incentivo, mesmo nos
momentos que as privei de minha companhia por estar
me dedicando à Universidade.*



Agradecimentos

Às Irmãs Salvat, Editeir (Booker), Cavalcanti, Neli e S. Pedro, todas pessoas de muita alta competência com quem tive a grata oportunidade de trabalhar tanto em outros projetos ligados a este trabalho. Ex-primos e primas que se tornaram meus amigos e que são de muita mais grande valia.

A Laila Gulyte S. Pereira, amiga e companheira, que com paciência e competência sempre me ajudou a finalizar o projeto do microprocessador que havíamos começado juntos, responsável por eu ter iniciado e concluído a minha linha de projetos de microprocessadores integrados.

Aos Irmãos E. de Lencastre, colega de trabalho e amigo, com quem tive a oportunidade de trabalhar e de quem aprendi muito sobre a tecnologia de microprocessadores e a importância da documentação para o sucesso de um projeto.

A DRA. MARIA Fátima Pereira e Prof. Dr. João Carlos L. L. Jr., representantes da CADENCE no Brasil, que me ajudaram durante o projeto, fornecendo sempre com qualquer tipo de suporte necessário para a realização do mesmo.

Este trabalho contou com o apoio da FUNDAÇÃO DE AMPARO À PESQUISA DO ESTADO DE S.PAULO - FAPESP, através dos processos 92/1121-8 e 95/6664-8



Agradecimentos

Ao Ito, Salvan , Eldemir (Becker), Cavalcanti, Neil e S. Pedro, todas pessoas da mais alta competência com quem tive a grata satisfação de trabalhar neste e em outros projetos ligados a este trabalho. Ex-alunos e orientados que se tornaram meus amigos e que são da minha mais elevada estima.

A Luís Otávio S. Ferreira, amigo e companheiro, que com sua calma e competência sempre me incentivou a continuar o projeto do microprocessador que havíamos começado juntos, responsável por eu ter iniciado a atuar na linha de projetos de microprocessadores integrados.

A Samuel E. de Lucena, colega de trabalho e amigo dos mais queridos, que me ajudou a revisar parte deste trabalho. As nossas discussões e suas sugestões serão fundamentais para dar continuidade ao mesmo.

A QUICKCHIP Engenharia e Projetos Eletrônicos Ltda, representante da CADENCE no Brasil, pelo apoio fornecido durante todo o projeto, procurando sempre sanar qualquer tipo de dúvidas em relação ao “software”, bem como assessorando-nos sempre que dela precisávamos.



Índice

Apresentação	1
Capítulo I Evolução dos Microprocessadores	3
A História dos Microprocessadores.....	3
Capítulo II Proposta do Dispositivo.....	8
II.1 - Introdução	8
II.2 - Arquitetura	10
II.3 - Características do Sistema a ser Controlado	12
II.3.1 - O Sistema de Elevadores.....	13
II.3.2 - Blocos Internos Especiais a Serem Implementados.....	15
II.3.3 - Lógica de Atendimento do Sistema	16
Capítulo III A Palavra Código e o Conjunto de Instruções:	21
III.1 - A Palavra Código	21
III.2 - O Conjunto de Instruções	22
Capítulo IV A Arquitetura Interna do Microcontrolador.....	32
IV.1 - Introdução.....	32
IV.2 - Máquina de Busca	32
IV.3 - Máquina de Execução.....	34
IV.4 - Máquina de Armazenamento	36
IV.5 - Gerenciador de Acesso	37
IV.6 - Arquitetura Proposta para o Microcontrolador	38
Capítulo V Simulação do Microcontrolador em Linguagem Verilog-HDL	40
V.1 - Introdução	40
V.2 - A Simulação Digital	40
V.3 - Circuito Preliminar - “Flip-Flop” Tipo D	42
V.4 - Máquina de Armazenamento.....	44



V.5 - Gerenciador de Acesso aos Barramentos.....	48
V.6 - Máquina de Busca	49
V.7 - Máquina de Execução.....	52
V.8 - Circuito de Controle	55
Capítulo VI Implementação do Circuito Desenvolvido	59
VI.1 - Considerações Iniciais	59
Capítulo VII Comentários Finais e Conclusão	62
Anexo 1 - Descrição dos Circuitos em Linguagem Verilog - HDL e Estímulos	64
Simbologia Utilizada.....	65
Circuito do Gerenciador de Acesso aos Barramentos	66
“Flip-Flop” utilizado na descrição das máquinas de estado	68
Máquina de Busca	70
Máquina de Execução.....	73
Máquina de Armazenamento	76
Circuito de Controle do Microcontrolador	78
Referências Bibliográficas.....	81



Apresentação

A indústria de aparelhos e sistemas eletrônicos tem feito uso, cada vez de modo mais constante, de circuitos integrados que trazem uma solução definitiva para os problemas por ela enfrentado em seus projetos. Estes dispositivos, que são encomendados aos projetistas e fabricantes unicamente para servir ao propósito único e exclusivo de quem os encomendou, são conhecidos como ASIC (Application Specific Integrated Circuits). Até recentemente, o universo dos ASICs se restringia aos circuitos analógicos, a uns poucos circuitos digitais ou mistos. Mais recentemente surgiu o conceito dos ASIPs - (Application Specific Instruction-set Processor), que são microprocessadores com um jogo de instruções especialmente elaborado para realizar uma determinada função, ou seja, um microprocessador dedicado.

O presente trabalho tem a proposta de apresentar não só o projeto e o desenvolvimento de um dispositivo ASIP, um microcontrolador integrado de 32 bits em filosofia RISC. Ele, também, tem a pretensão de mostrar que, com os recursos existentes de ferramentas computacionais, tanto de "hardware", como de "software", é possível desenvolver projetos de elevado grau de sofisticação, em ambientes universitários, de modo que este centro de pesquisa, que é a Universidade, venha contribuir de maneira significativa para a diminuição da distância tecnológica verificada entre o nosso país e os de primeiro mundo. Neste trabalho, não se teve a intenção de obter um dispositivo microcontrolador completo, na forma integrada, mas mostrar que, caso um grupo deseje desenvolver um dispositivo semelhante, isto é possível de ser feito. Deste modo, ele inicia com a definição de um problema prático, o controle de um grupo de elevadores, passando para a escolha da filosofia de sua arquitetura, a definição da palavra e do jogo de instruções, a elaboração da arquitetura interna do dispositivo, simulação, em linguagem de descrição de "hardware" dos blocos de seus "pipelines", discussão dos resultados obtidos; escolha da tecnologia e forma de integração.

Este trabalho, também, tem a intenção de contribuir como um roteiro para aqueles que desejarem iniciar o desenvolvimento de projetos semelhantes, na área de microprocessadores, que assim como outras áreas da microeletrônica é tão carente de recursos humanos. Deste modo, procurou-se apresentá-lo na mesma seqüência que foi utilizada no desenvolvimento do dispositivo, e da forma mais didática possível.

Evolução dos Microprocessadores

A História dos Microprocessadores

Dentre as descobertas desta segunda metade do século XX, talvez a mais importante tenha sido a da tecnologia dos semicondutores e, mais especificamente, a dos circuitos integrados. O impacto desta descoberta pode ser observado no aumento da capacidade de armazenamento de dados e no aumento da velocidade de processamento. A história dos microprocessadores é a história da evolução da tecnologia dos semicondutores. Desde a década de 1950, os circuitos integrados passaram por várias gerações, cada uma com suas características próprias. A primeira geração, conhecida como tecnologia de silício, foi substituída pela segunda, a tecnologia de silício bipolar, e esta, por sua vez, foi substituída pela terceira, a tecnologia de silício MOSFET. A quarta geração, a tecnologia de silício CMOS, é a mais recente e a mais utilizada atualmente.

- Novembro de 1971: A Intel apresenta o primeiro microprocessador, o Intel 4004.
- 1972: A Intel apresenta o segundo microprocessador, o Intel 8008.
- 1974: A Intel apresenta o terceiro microprocessador, o Intel 8080.
- 1976: A Intel apresenta o quarto microprocessador, o Intel 8085.
- 1978: A Intel apresenta o quinto microprocessador, o Intel 8086.
- 1982: A Intel apresenta o sexto microprocessador, o Intel 286.
- 1985: A Intel apresenta o sétimo microprocessador, o Intel 386.
- 1989: A Intel apresenta o oitavo microprocessador, o Intel 486.
- 1993: A Intel apresenta o nono microprocessador, o Intel Pentium.
- 1995: A Intel apresenta o décimo microprocessador, o Intel Pentium Pro.
- 1997: A Intel apresenta o décimo primeiro microprocessador, o Intel Pentium II.
- 1999: A Intel apresenta o décimo segundo microprocessador, o Intel Pentium III.



Capítulo I

Evolução dos Microprocessadores

A História dos Microprocessadores

Dentre as descobertas desta segunda metade do século, entre as mais importantes, certamente, está a da tecnologia que possibilitou o desenvolvimento do transistor, inicialmente, e posteriormente do circuito integrado. O impacto deste desenvolvimento pode ser apreciado na influência exercida, pelo uso destes dispositivos, no dia-a-dia das pessoas. As facilidades proporcionadas são muito maiores do que poderiam conceber inicialmente as cabeças mais imaginativas. Dentro da área voltada para os circuitos integrados, um dos mais significativos desenvolvimento pode ser atribuído ao microprocessador. Uma rápida revisão da sua história e das datas mais importantes que envolve a evolução deste circuito, segundo [1], é feita a seguir.

- **Novembro de 1971:** A Intel noticiou seu primeiro microprocessador;
- **1972:** A Intel cumpriu o prometido com entrega do microprocessador 8008;
- **1974:** É colocado à disposição o microprocessador 8080;
- **1976:** Ocorre o lançamento do microprocessador 8085;
- **1978 :** É posto em disponibilidade o microprocessador 8086;
- **1983-1984:** Os microprocessadores da linha 80286 são colocados no mercado;
- **1986:** A Intel lança o microprocessador 80386;
- **1990:** É introduzido o microprocessador 80486;



- 1992: É lançado o microprocessador PENTIUM.

Durante este intervalo, o desempenho dos mesmos melhorou em mais de trezentas vezes, tendo, em contrapartida, o preço deste dispositivo diminuído, em média, de \$500,00 para \$5,00. A melhoria do desempenho pode ser atribuída diretamente à evolução tecnológica. Hoje já é possível confeccionar dispositivos com tecnologia CMOS com comprimento de canal da ordem de $0,25\mu\text{m}$ [2], além de memórias com capacidade de 4Gb [3] e tempos de acesso da ordem de 46ns [4]. A diminuição da tensão de polarização dos dispositivos foi fundamental para o aumento da velocidade dos microprocessadores, chegando mesmo, a encontrar na literatura, o anúncio de dispositivos com frequência de clock em torno de 600 MHz [5-6]. Os microcomputadores entraram em nossos lares, escritórios, e automóveis, modificando conceitos e expectativas, ficando cada vez mais evidente a sua importância dentro dos sistemas eletrônicos e mostrando que vieram para ficar.

Em 1969, engenheiros da Intel começaram a estudar meios de integrar e particionar as funções de controle de calculadoras e tubo de raios catódicos em dispositivos LSI. Ao mesmo tempo, outras companhias (principalmente a Texas Instruments) procuraram meios de reduzir o tempo necessário para desenvolver projetos de circuitos integrados. Naquele ano a Intel iniciou dois projetos patrocinados por seus clientes: o 4004, para calculadoras, e o 8008, para controle de tubo de raios catódicos (TRC). O dispositivo 4004 viria a substituir seis dos circuitos de um sistema desenvolvido pelo contratante. Em relação às suas arquiteturas, tanto o 4004, como o 8008 são considerados “unidades centrais de processamento”, conhecidas como CPUs, completas, em um único dispositivo e possuem características semelhantes, mas, em função do 4004 ter sido projetado para uso em um sistema de aritmética serial em BCD e o 8008 para manipulação de 8 bits, seus jogos de instrução são diferentes.

A Datapoint, inicialmente conhecida como “Computer Terminal Corporation”, contratou a Intel para projetar um dispositivo que viria a ser usado em um terminal de TRC. A Datapoint pretendia, também, construir um processador de bit-serial em lógica de TTL, utilizando memórias que faziam uso de registradores de deslocamento. A Intel propôs que o processador inteiro poderia ser implementado em um único dispositivo. Este processador viria a se tornar o



8008 que, junto com o 4004, foram fabricados utilizando tecnologia PMOS, tecnologia esta usada até então em fabricação de memórias.

A Datapoint especificou a maioria do jogo de instrução e a organização dos registros. A Intel modificou este jogo de instrução de modo que o processador pudesse ser integrado em um único dispositivo e acrescentou instruções adicionais para tornar o 8008 um dispositivo de propósito geral.

A arquitetura do 8008 é bem simples, quando comparada com os microprocessadores de hoje. O espaço de memória é limitado a 16 kilobytes, e a pilha (pequena área de memória para o armazenamento temporária de dados) fica interna ao dispositivo, armazenando, porém, somente oito linhas de dados. O jogo de instrução é pequeno, com somente alguns modos de endereçamento disponível.

Por volta de 1973, a tecnologia de fabricação de memórias tinha avançado de PMOS para NMOS. Como um exercício de engenharia, a Intel usou jogo de máscaras do 8008 em um processo NMOS para obter maiores velocidades de operação. Depois de um curto estudo, determinou que seria necessário um novo "layout" para as máscaras. A companhia decidiu, também, melhorar o processador e ao mesmo tempo utilizar um novo encapsulamento de 40 pinos. O resultado foi o processador 8080.

O 8080 foi o primeiro processador especificamente projetado para o mercado de microprocessadores. O dispositivo foi projetado para incluir todo o jogo de instruções do 8008, embora não necessariamente com os mesmos códigos de operação. Em relação ao 8008, o 8080 teve uma melhora de 10 para 1, em desempenho, eliminando muitas das falhas verificadas nos 8008, além da obtenção de uma capacidade de processamento superior a encontrada anteriormente. Como melhorias pode-se citar a capacidade de manipulação de dados de 16 bits (principalmente para cálculos), aritmética BCD, melhoria nos modos de endereçamento, e melhoria no processamento das interrupções.

Por volta de 1976, o avanço tecnológico permitiu a Intel considerar implementação de melhorias nos microprocessadores 8080. Entre os vários pontos falhos identificados nos dispositivos 8080 estão a exigência do uso de uma fonte tripla de alimentação e de um dispositivo oscilador. A idéia



era produzir um processador que operasse com uma única fonte de alimentação e exigisse menos dispositivos de apoio. O processador resultante foi chamado de 8085 e teve a intenção de ser uma máquina com código compatível com os 8080. Assim, o jogo de instrução do 8080 poderia ser quase que totalmente utilizado com o 8085.

Os microprocessadores da linha 8086, um processador de 16 bits de dados, foi projetado para proporcionar um aumento de uma ordem de magnitude sobre ambos, o 8080 e o 8085. O processador foi idealizado para ser compatível com os 8080 e 8085 em relação à sua linguagem “assembly”, de modo que os “softwares” existentes dos dispositivos 8080 e 8085 podiam ser compilados novamente e executados no 8086.

Este resumo histórico utilizou a linha Intel para exemplificar a evolução do microprocessadores por ter sido esta companhia a primeira a lançar este dispositivo no mercado, porém, fabricantes como a Zilog e a Motorola não podem deixar de serem citados.

A Zilog produziu o Z80, talvez o mais utilizado microprocessador de 8 bits. O Z80, uma versão melhorada do 8080, é compatível com este, isto é, pode executar os códigos escritos para os 8080, além de possuir as suas próprias instruções adicionais. Este dispositivo fez e ainda faz um enorme sucesso com os projetistas de sistemas.

A Motorola, outra pioneira na área de semicondutor, possui uma completa linha de microprocessadores de 8 e 16 bits. A família Motorola 68000 de microprocessadores oferece várias vantagens sobre o processador 8086. O Motorola 68020 é um microprocessador verdadeiro de 32 bits. Este processador tem 16 registros de 32 bits de propósito geral, um contador de programa de 32 bits, um registrador de “status” de 16 bits, além de outras melhorias. O 68020 pode manipular bit, byte, 16-bits, 32 bits e operandos de 64 bits. A capacidade de endereçamento direto deste dispositivo é de quatro gigabytes.

Com o aumento do número de aplicações, da demanda de uso e de exigências de desempenho, novos dispositivos foram desenvolvidos e novas arquiteturas concebidas procurando satisfazer as necessidades crescentes dos engenheiros e projetistas de sistemas. Arquiteturas do tipo RISC (Reduced Instruction Set Computer), WISC (Writable Instructions Set Computer e VLIW (Very



Long Instruction Word Computer), trouxeram conceitos novos, sempre objetivando o aumento de desempenho dos mesmos.

Com a evolução das arquiteturas, começaram a surgir microprocessadores dedicados a grandes áreas. Deste modo, pode-se citar o desdobramento destes dispositivos em microcontroladores e DSP (Digital Signal Processor), além daqueles dedicados ao processamento de dados. Fundamentalmente, estas ramificações nada mais são dos que os conhecidos microprocessadores, porém, com uma arquitetura dedicada para cada especialidade. Os microcontroladores possuem a sua arquitetura e seu jogo de instruções dedicados à funções de controle não sendo difícil encontrar nos mais recentes lançamentos, dispositivos com conversores A/D , D/A e moduladores PWM internos a eles. Os DSPs, por sua vez, possuem uma imensa capacidade de processamento aritmético, tão necessário no momento de se implementar filtros. Os autores da referência [7] fazem uma comparação bastante detalhada destas arquiteturas.

Mais recentemente, com o desenvolvimento de novas e poderosas ferramentas dedicadas à projeto de circuitos integrados, os projetistas de sistemas eletrônicos puderam pleitear o projeto de um dispositivo dedicado que satisfizesse a todas as suas especificações, dentro de uma finalidade específica. Esta classe de circuitos, que inicialmente foram denominados ASICs (Application Specific Integrated Circuits), teve uma ramificação quando o dispositivo dedicado a ser projetado se tratasse de um microprocessador. A este microprocessador dedicado passou-se a chamar ASIP (Application Specific Instruction-Set Processor).



Capítulo II

Proposta do Dispositivo

II.1 - Introdução

A proposta deste trabalho é o desenvolvimento de um dispositivo microcontrolador para um uso dedicado. Esta classe de dispositivos foi recentemente classificada como ASIP (Application Specific Instructions-set Processor) ou “processadores com jogo de instruções para uma aplicação específica”. Esta denominação de dispositivos pode ser enquadrada dentro de uma classe maior de circuitos integrados para uso dedicado denominada ASICs (Application Specific Integrated Circuits) ou “circuitos integrados para uma aplicação específica”.

Recentemente tem-se presenciado um crescimento bastante grande no desenvolvimento de circuitos especialmente projetados para uso em uma aplicação pré-definida. Este crescimento é verificado não somente em função do avanço das ferramentas de projeto, o que certamente colaborou em muito para que isto acontecesse, mas , também, em função de outros fatores inerentes ao uso destes dispositivo pelos projetistas de sistemas, tais como:

- *Aumento da confiabilidade do sistema projetado;* O aumento da confiabilidade encontra-se diretamente relacionado, numa placa, com o número de componentes que nela são colocados. Quanto menor o número de componentes, menores são as chances de ocorrerem problemas. O uso de ASIC diminui sensivelmente o número de dispositivos exigidos na montagem de um sistema, em relação ao uso de dispositivos comerciais.



- *Capacidade de reduzir o espaço requerido na placa;* A capacidade redução da placa encontra-se relacionada com a densidade de componentes para a montagem do sistema projetado. Usar ASICs em uma placa libera espaço que normalmente seria utilizado por diversos componentes comerciais necessários na montagem para a realização da lógica projetada.
- *Segurança em relação a cópias do sistema;* A dificuldade de se copiar um circuito integrado é muitas ordens de grandeza maior do que a cópia de uma placa de circuito impresso. O fabricante do sistema terá uma maior segurança em relação aos seus concorrentes quando da colocação de seu sistema no mercado. Um outro fato que pode ser ressaltado é que somente ele possui os componentes para reposição para o caso da manutenção. Este fato estratégico é de fundamental importância para o fabricante.
- *Maximização da performance;* A maximização da performance de um ASIC está vinculada às vantagens citadas acima: reduzir as partes e aumentar a disponibilidade de espaço na placa permitem ao projetista colocar mais funções numa mesma placa.

Em função do alto uso de dispositivos programáveis, e das inerentes vantagens de se usar dispositivos ASICs, foi desenvolvida uma linha de pesquisa e projetos dentro desta classe de dispositivos, procurando atender as necessidades daqueles que usam dispositivos dedicados voltados para processamento de dados e de sinais e controle.

Em relação aos dispositivos programáveis, os microcontroladores são dispositivos eletrônicos semelhantes aos já conhecidos microprocessadores, porém com a sua arquitetura especialmente projetada para tarefas de gerenciamento e controle industrial. É comum estes dispositivos já contarem internamente com memórias, conversores A/D e D/A, e circuitos de I/O, dentre outros. Em função disto, do seu custo, e da flexibilidade que estes dispositivos oferecem, os locais onde podem ser aplicados se estende cada dia mais. Hoje, encontram-se microcontroladores não somente na indústria, em controle de processos, mais também com aplicações na área automotiva, na área de instrumentação e em produtos de consumo como brinquedos e eletrodomésticos. Em suma, este dispositivo veio substituir as caras e onerosas placas contendo diversos CIs, de um modo simples e elegante [13].



O trabalho aqui apresentado propõe o desenvolvimento de um microcontrolador voltado para atuar em um sistema de elevadores de atendimento inteligente. Seguindo essa idéia, foram levantadas características gerais e funcionais desse sistema, foi definido um conjunto de instruções direcionado para a realização da tarefa proposta, e baseado nesse jogo de instruções, desenvolveu-se a arquitetura interna do dispositivo. Optou-se pela filosofia de processamento RISC (Reduced Instruction Set Computer), em que se destacam características tais como alta velocidade de processamento, um jogo de instruções reduzido, e uma maior facilidade de implementação de seu circuito, além de possuir uma estrutura altamente regular.

II.2 - Arquitetura

Para o desenvolvimento deste trabalho, adotou-se a filosofia RISC (Reduced Instruction Set Computer) para a arquitetura do microcontrolador. Os motivos que levaram a esta escolha são fáceis de serem justificados quando se compara as facilidades conseguidas no projeto do dispositivo com esta filosofia de arquitetura em relação a projetos que utilizam filosofia CISC (Complex Instruction Set Computer).

Os jogos de instrução e seus modos de endereçamento, em um projeto de um microprocessador, tendem a aumentar em número e complexidade. Como exemplo, pode-se citar o minicomputador 11/780 da VAX, que tem 16 modos de endereçamento e mais de 300 instruções [1]. Mesmo microprocessadores comerciais possuem jogos de instruções complexos. O Motorola 6802, por exemplo, reconhece sete tipos de dados, emprega 18 modos de endereçamentos, além de possuir formatos de instruções que podem variar no seu comprimento, de 1 palavra de 16 bits até 11 palavras de 16 bits. O uso de um jogo complexo de instruções requer que a “unidade central processamento”, mais conhecida como CPU, tenha também uma complexa unidade de controle decodificadora de instrução. Na família Motorola 68000 de microprocessadores, a unidade de controle ocupa mais de 60% da área total do dispositivo. Esta atribuição representa uma porção significativa dos recursos de silício na CPU, além de um número maior de homens/hora para o seu projeto.

Um desenvolvimento relativamente recente em projeto de arquitetura de CPU é uma filosofia que, cuidadosamente, adapta o jogo de instrução às necessidades mais imediatas de uso desta CPU,



fazendo com que as instruções mais utilizadas estejam presentes no seu jogo de instruções. As menos utilizadas são sintetizadas, a partir destas instruções primitivas, quando delas se necessitar. Para isto deve-se utilizar, durante a fase de elaboração de seus programas, compiladores que facilitem este serviço.

A implementação destas poucas instruções "essenciais" é executada de modo que o arranjo da estrutura interna do circuito integrado fique o mais regular possível, particularmente o da unidade de controle. Este arranjo regular, além de facilitar o projeto do dispositivo, leva ao uso cada vez mais otimizado das poderosas ferramentas de projeto de circuitos integrados encontradas hoje em dia. Em contraste com a filosofia CISC, arquiteturas RISC utilizam jogo de instruções cuidadosamente escolhido para criar e projetar as CPUs, o que resulta em sistemas com um alto poder de processamento e uma menor relação custo/desempenho.

Os conceitos relativos à arquitetura RISC não são particularmente novos. A sua implementação, no entanto, teve que esperar o avanço e o desenvolvimento de tecnologias voltadas para dispositivos VLSI. A referência [1] apresenta um bem detalhado histórico desta filosofia.

Em geral, a definição de uma arquitetura como sendo RISC é freqüentemente utilizada. Porém, somente o número de instruções não é um fator adequado para julgar o tipo de arquitetura que um processador possui. Fundamentalmente, existem atributos básicos para definir uma máquina como sendo RISC. Estes atributos são:

1. A arquitetura de RISC tem um número relativamente baixo de instruções primitivas ou básicas, preferivelmente menos que 100. Por exemplo, o RISC I [14] tem 31 instruções; o RISC II [15] tem 39 instruções; o MIPS de Stanford [1] tem 31 instruções. Todas as outras instruções são sintetizadas a partir das instruções originais. Além do mais, utiliza-se um grande número de operações do tipo registro-a-registro. Esta restrição simplifica significativamente o projeto do "hardware" de uma máquina RISC.
2. Existe um número relativamente baixo de modos endereçamentos, normalmente um ou dois.



3. Há um número relativamente baixo de formatos de instrução, normalmente um ou dois. As instruções são de comprimento fixo.
4. Todo acesso à memória é executado por instruções de "load"/"save" (Carga/Salvamento) e instruções de desvio e manipulação de pilha.
5. Cada instrução *praticamente* é executada em um ciclo do relógio de sistema. Isto é, a medida de "relógios por instrução" (CPI - Clock per Instructions) fica perto da unidade.
6. Existe um número relativamente grande de registros na CPU. Tipicamente o número é maior que 32. Deste modo, onde for possível, a operação registro-a-registro é realizada no lugar de operações memória a registro.
7. A unidade de controle é feita por hardware, no lugar do uso de "software" e micro-instruções. Esta simplicidade favorece o aumento de sua velocidade.

II.3 - Características do Sistema a ser Controlado

Sistemas de elevadores como os tradicionais contam, fundamentalmente, com circuitos de comando baseados em relés eletromecânicos (contatores), cujas características principais são a complexidade e necessidade de constante manutenção (preventiva e corretiva). A proposta deste trabalho é de desenvolver um dispositivo eletrônico, dedicado para o controle do processo em questão, com um jogo de instruções apropriado, de modo a melhorar a eficiência e confiabilidade de um sistema de elevadores. Porém, o dispositivo idealizado poderá servir, também, para uso em outros sistemas de controle diferentes do apresentado aqui. Para tanto, tomaram-se cuidados adicionais, quando de sua concepção, para satisfazer esta exigência. Deste modo, esse dispositivo possibilitará a implementação de diversos e diferentes sistemas, de acordo com a propósito ou objetivo de seu projetista. Isto é conseguido de forma relativamente simples, uma vez que o dispositivo contará com blocos básicos e específicos para controle de processos. A utilização adequada de cada um destes blocos é que tornará possível a diversificação de seu uso



A seguir, será apresentado o sistema de elevadores idealizado e hipotético, onde estão colocadas as principais características de controle que se deseja obter, bem como a lógica de atendimento às chamadas realizadas pelos usuários. A especificação deste sistema de elevadores, neste momento, torna-se necessária em função do dispositivo proposto ser um ASIP, com um conjunto de instruções e a arquitetura interna totalmente definidas para este sistema.

II.3.1 - O Sistema de Elevadores

Para a definição do sistema a ser controlado foram levantadas algumas das principais características exigidas de um sistema tradicional de elevadores, bem como definidas algumas novas possíveis de serem implementadas com o microcontrolador idealizado. As características aqui apresentadas são um exemplo de um sistema de elevadores genérico. Deste modo, como características principais o sistema proposto terá:

-Prioridade de atendimento para andares específicos:

Alguns andares, se programados, podem ter prioridade de chamada, ou seja, a cabine do elevador se deslocará imediatamente ao referido andar, mesmo existindo chamadas pendentes em outros.

-Programação do número de andares e cabines:

O número de andares e cabines do sistema de elevadores é pré-definido no seu projeto, tanto de “software”, como de “hardware”. No caso do sistema sugerido, supõe-se que existam mais de 3 andares e mais de uma cabine paralela por andar. Desta forma, procura-se atingir um caso mais genérico, de onde possam surgir diferentes propostas de sistemas. Existe, porém, um nível máximo suportável pelas características do dispositivo. Esse nível máximo será de 16 andares, para um número de cabines, em primeira análise, ilimitado.

-Operação automática ou manual (ascensorista):

Esta função também deve ser pré-definida no desenvolvimento do projeto. O sistema sugerido implementa o modo automático.



-Alarme para excesso de carga:

O sistema ficará sempre atento ao limite máximo de carga de cada cabine. Caso o limite seja atingido ou ultrapassado, o sistema bloqueará o atendimento às chamadas até que a cabine sofra um decréscimo de carga e volte a um nível abaixo do limite. Esse limite, no entanto, não é o máximo suportado pela cabine, uma vez que acionado o alarme, o sistema deverá continuar operando, num regime de sobrecarga, por um período de tempo reduzido.

-Controle de mais de um elevador simultaneamente com lógica inteligente de atendimento às chamadas:

O sistema sugerido é formado por mais de uma cabine paralela. Desta forma, propõe-se o atendimento inteligente às chamadas externas: apenas uma cabine será enviada ao andar da chamada. A cabine enviada será a que proporcionar o menor tempo de espera para o usuário.

Obs:

- 1) Chamadas externas são aquelas realizadas pelos usuários nos andares de atendimento. Chamadas internas indicam o andar de destino desejado pelo usuário que está dentro da cabine do elevador.
- 2) Em cada andar existirão dois botões para chamadas externas, independente do número de cabines. O usuário deverá pressionar o primeiro botão se o seu andar de destino for para cima, e o segundo botão, se for para baixo.

É interessante ressaltar mais uma vez que tais características podem ser excluídas e outras novas implementadas, de acordo com as necessidades e objetivos que se queira dar ao projeto. A arquitetura do microcontrolador, permite essa gama de variações, trazendo facilidades e maior flexibilidade ao projetista do sistema.



II.3.2 - Blocos Internos Especiais a serem Implementados

Em dispositivos microcontroladores é comum a sua arquitetura abrigar um arquivo de registros de uso geral e alguns registros especiais. Para o sistema aqui proposto, observou-se a necessidade da implantação de três registros específicos, além de portas adicionais de entrada e saída. Deste modo tem-se:

Registros de Chamadas Pendentes e de Destinos

Quando da ocorrência de uma chamada externa ou interna, o andar de onde provém a chamada externa ou o andar de destino da chamada interna, é inserido num registro de chamadas pendentes. Esse andar ficará aí registrado até que a lógica do sistema defina o seu atendimento por determinada cabine. Neste ponto, o andar é retirado do registro de chamadas pendentes e colocado no registro de destinos da cabine, a qual efetuará o atendimento. Portanto, cada cabine estará relacionada a um arquivo de destino, onde estarão os andares de parada da cabine específica. É interessante afirmar aqui que, no caso de uma cabine se encontrar numa trajetória de subida, somente serão relacionados a ela andares de destino que estiverem acima da sua posição atual. Desta forma, a cabine realizará as paradas segundo uma simples comparação entre sua posição atual e os andares contidos em seu registro de destinos. Para facilitar a lógica do sistema implementou-se um registro de chamadas pendentes de subida e outro de descida. Desta forma, os três registros específicos serão:

- Registro de chamadas pendentes de subida;
- Registro de chamadas pendentes de descida;
- Registro de destinos.

Obs: Cada cabine estará relacionada a um registro de destinos individual, enquanto os registros de chamadas pendentes serão compartilhados por todo o sistema (esses registros de chamadas pendentes poderiam até ocupar um espaço de memória externa, para facilitar o seu acesso por todo o sistema).



Portas de Entrada

Para a entrada das chamadas externas para inclusão no registro de chamadas pendentes, existem duas opções:

- 1) *Leitura multiplexada (por varredura dos andares)*: Este modo é econômico em relação ao número de pinos reservados ao dispositivo para esta função, possibilitando a implementação de sistemas de elevadores de muitos andares. Torna-se porém mais dispendioso em relação ao tempo de execução do “software” e complexidade da Arquitetura.
- 2) *Entrada simples de um bit por andar*: Este modo facilita a elaboração da Arquitetura do dispositivo e agiliza a execução do software. Porém, limita a capacidade máxima de andares atendidos pelo sistema.

Neste projeto foi adotada a segunda opção, uma vez que a mesma simplificará a arquitetura do dispositivo sem trazer prejuízos aos objetivos traçados. Sendo assim, resta agora definir a largura desta porta de entrada, o que determinará o máximo número de andares atendidos pelo sistema. Será adotada então uma porta de 16 bits.

Esta porta deverá ser compartilhada pelas entradas de Chamadas Externas de Subida e de Descida, e pela entrada da posição atual da cabine. Este compartilhamento da porta é realizado estabelecendo endereços para cada uma das entradas.

Portas de Saída

A arquitetura do microcontrolador deverá implementar uma porta de saída de 16 bits para o envio da informação de destino da cabine.

II.3.3 - Lógica de Atendimento do Sistema

Caso a cabine do elevador esteja subindo com destino a um determinado andar, e sejam realizadas chamadas em andares que estão na sua trajetória, estas chamadas também serão atendidas. A



cabine só irá parar de subir quando não houver mais chamadas para cima, ou se a cabine atingir o último andar. O mesmo raciocínio é válido quando a cabine estiver descendo.

O mecanismo para que isto aconteça é compartilhar o registro de chamadas pendentes, com a lógica de cada uma das cabines (em caso de existirem várias cabines). Desta forma, se obtém o atendimento inteligente de apenas uma das cabine, a cada chamada externa, uma vez que o atendimento a uma chamada é realizado com a retirada do andar do registro de chamadas pendentes para a sua introdução no registro de destinos de uma das cabines. A figura III.1 apresenta o fluxograma para a lógica de atendimento em questão. Esse fluxograma representa a lógica de atendimento para uma das cabines.

A implementação de um sistema de mais de uma cabine implicará no acréscimo de novos dispositivos microcontroladores. Neste caso, o registro de chamadas pendentes pode ser um espaço de memória externa, de acesso permitido a qualquer um dos dispositivos. O registro de destinos pode, no entanto, ser interno a cada microcontrolador.

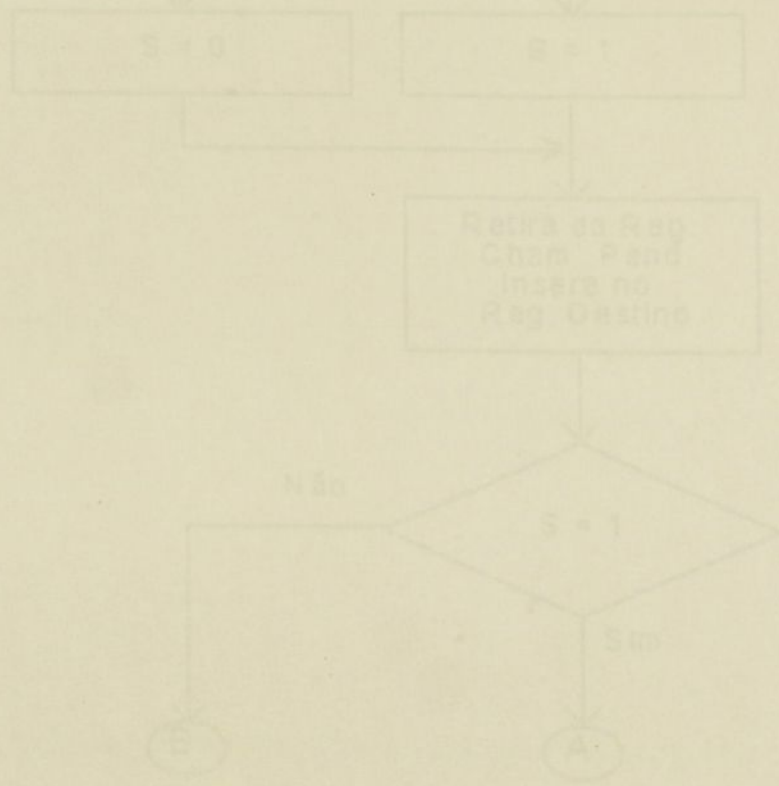


Figura III.1 - Fluxograma de atendimento para uma das cabines propostas

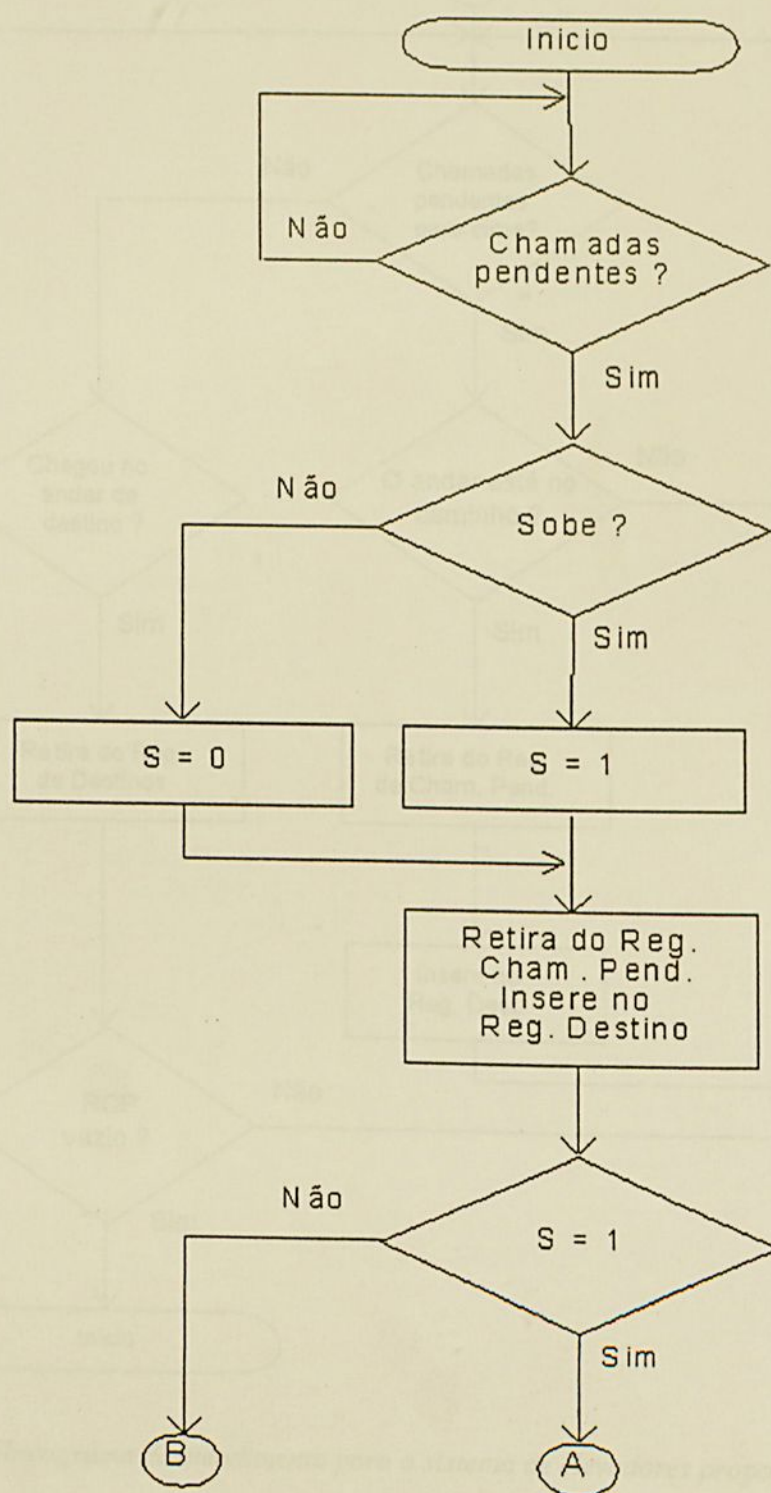


Figura III.1 - Fluxograma de atendimento para o sistema de elevadores proposto.

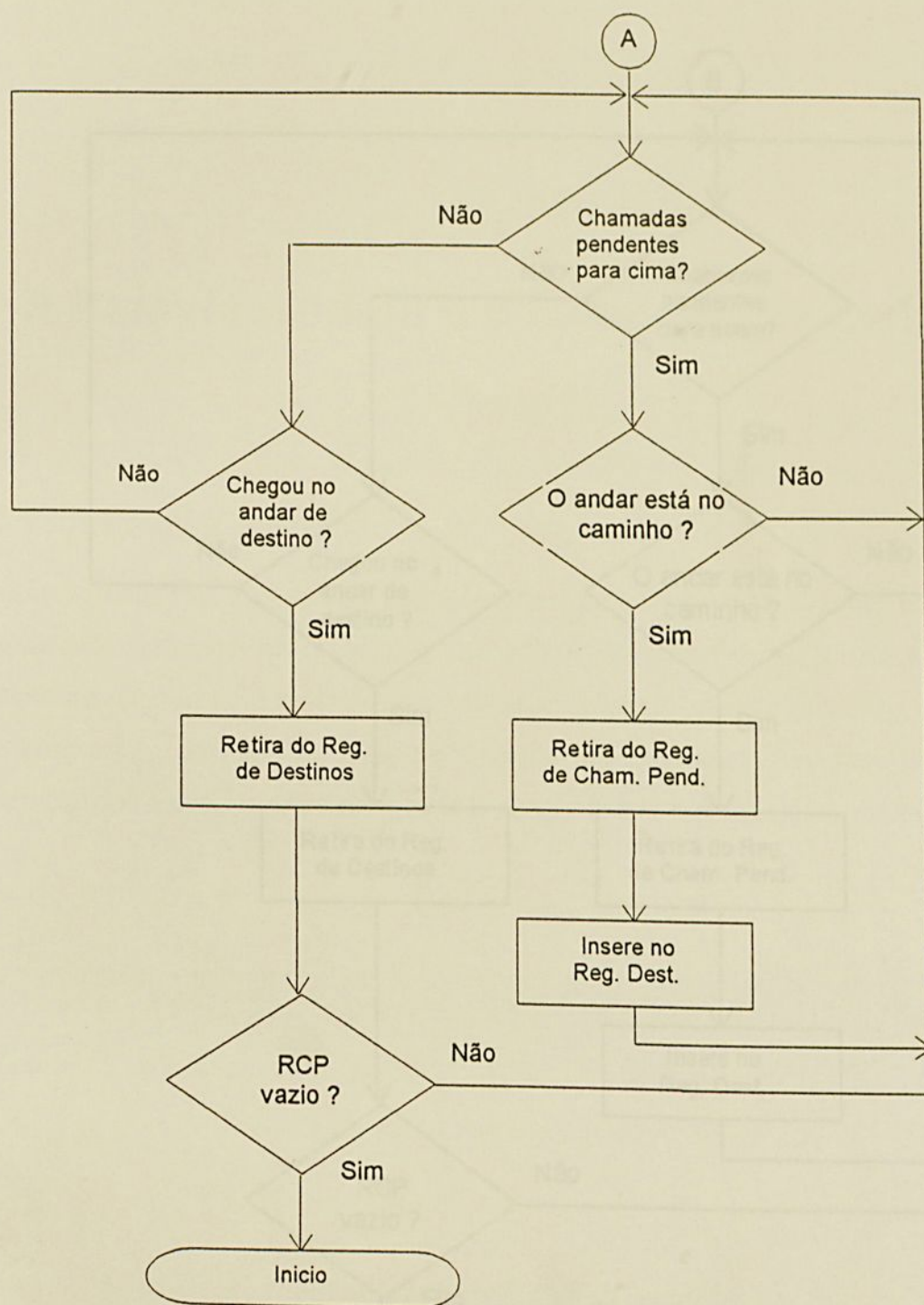


Figura III.1b - Fluxograma de atendimento para o sistema de elevadores proposto (continuação)

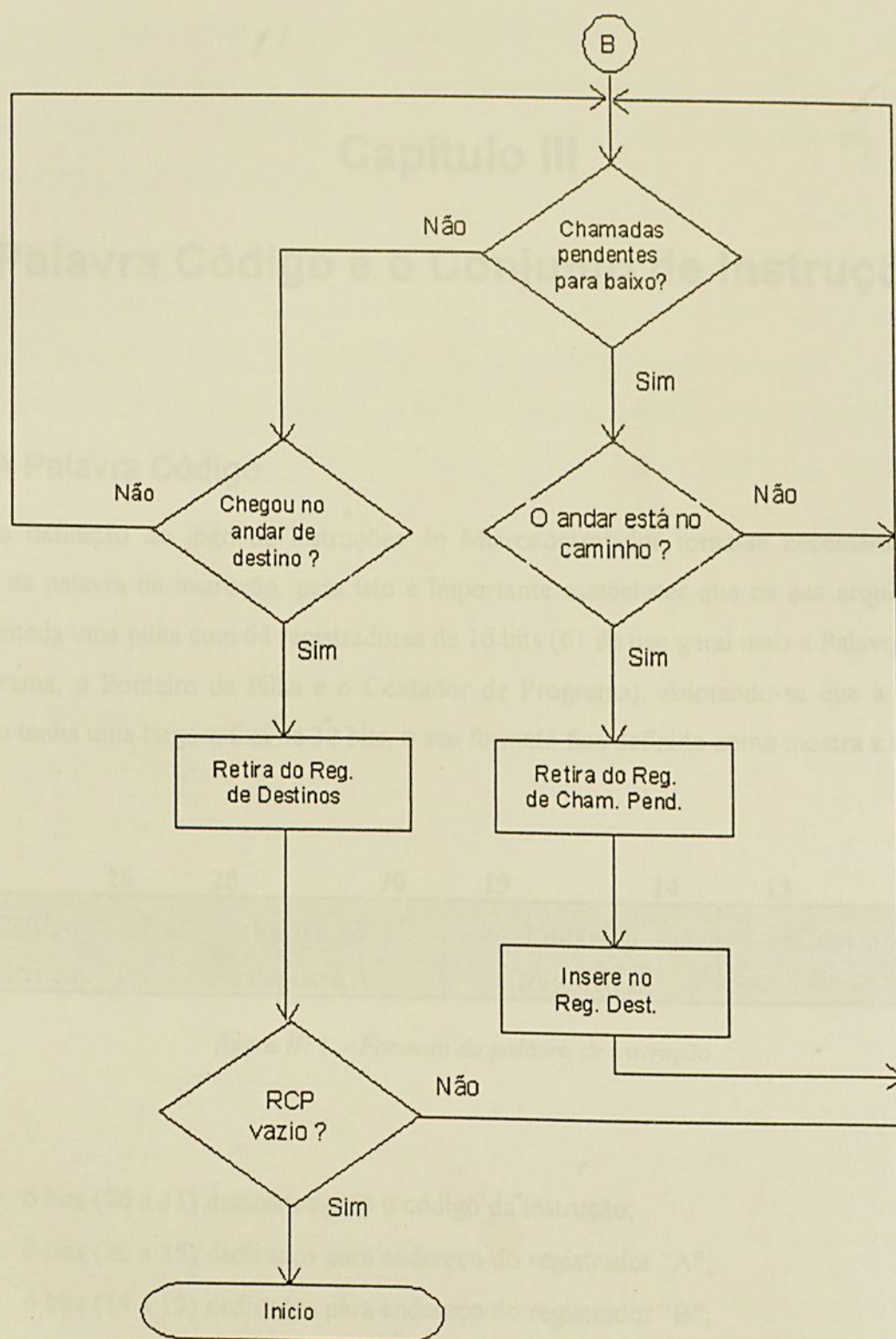


Figura III.1c - Fluxograma de atendimento para o sistema de elevadores proposto (continuação)

Capítulo III

A Palavra Código e o Conjunto de Instruções:

III.1 - A Palavra Código

Antes da definição do jogo de instruções do Microcontrolador torna-se necessário definir o formato da palavra de instrução, para isto é importante estabelecer que na sua arquitetura será implementada uma pilha com 64 registradores de 16 bits (61 de uso geral mais a Palavra de Status de Programa, o Ponteiro da Pilha e o Contador de Programa). Adotando-se que a palavra de instrução tenha uma largura fixa de 32 bits, o seu formato fica definido como mostra a figura III.1

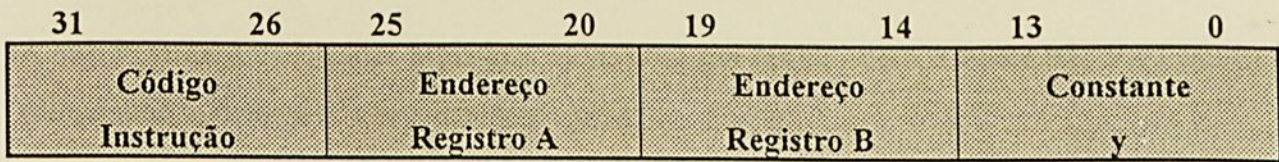


figura III.1 - Formato da palavra de instrução

sendo:

- 6 bits (26 a 31) dedicados para o código da instrução;
- 6 bits (20 a 25) dedicados para endereço do registrador "A";
- 6 bits (14 a 19) dedicados para endereço do registrador "B";
- 14 bits dedicados para uso de uma constante "y", quando a instrução assim o exigir.



III.2 - O Conjunto de Instruções

O jogo de instruções necessário para a definição da arquitetura do dispositivo proposto foi definido a partir de algumas das características observadas nos microcontroladores encontrados no mercado (8031 e 8097BH) e das características específicas definidas para este microcontrolador, conforme aqui apresentado.

As instruções definidas para o dispositivo proposto são de dois tipos: *do tipo básicas*, onde são contempladas as exigências de um microcontrolador com as características de um dispositivo que possui uma arquitetura RISC; e *instruções especiais*, definidas especificamente para cumprir a proposta deste dispositivo. Apesar disto, possibilitará o desenvolvimento de projetos de controle dos mais diferentes tipos, podendo este dispositivo vir a ser utilizado nas mais diferentes aplicações.

Para a definição das instruções as mesmas foram divididas em três grupos, de acordo com o seu tipo:, sendo:

- 1) Transferência de dados;
- 2) Saltos e desvios;
- 3) Operações entre registros;

A seguir serão apresentadas estas instruções com um resumo de sua operação. Para facilitar o entendimento, definiu-se a seguinte simbologia:

y = constante;
Rd = Registrador destino;
Rf = Registrador fonte;
Ga = Porta I/O;
PC = Contador de programa;
@ = condicional “se”;
= O bit do “ “ do registrador “ “;
SP = Ponteiro da pilha;
>> = Rotação à direita;
<< = Rotação à esquerda



a) Instruções de Transferência de Dados:

1) LDM (Carrega dado da memória):

Sintaxe: LDM Rd,y

Operação: Rd (y)

Resumo: o valor y endereça uma localização de memória, cujo conteúdo será copiado para o registro Rd.

2) LDR (Carrega dado da memória endereçada por registro):

Sintaxe: LDR Rd,RfB

Operação: Rd (RfB)

Resumo: o valor contido no registro RfB endereça uma localização de memória, cujo conteúdo será copiado para o registro Rd.

3) LDI (Carrega dado imediato):

Sintaxe: LDI Rd,y

Operação: Rd y

Resumo: o valor y é carregado imediatamente no registro Rd.

4) LDG (Carrega dado da porta de E/S):

Sintaxe: LDG Rd,Ga

Operação: Rd (Ga)

Resumo: o valor Ga endereça uma porta E/S, carregando o seu conteúdo no registro Rd.

5) STM (Grava dado na memória):

Sintaxe: STM RfA,y

Operação: (y) RfA

Resumo: O conteúdo do registro RfA é carregado na posição de memória endereçada por y.



6) STR (Grava dado na memória endereçada por registro):

Sintaxe: STR RfA,RfB

Operação: (RfB) RfA

Resumo: a locação de memória, endereçada pelo conteúdo do registro RfB, recebe o valor contido em RfA.

7) STG (Grava dado na porta de E/S):

Sintaxe: STG RfA,Ga

Operação: (Ga) RfA

Resumo: o conteúdo do registro RfA é enviado a porta de E/S e endereçada pelo valor Ga.

b) Instruções de Saltos e Desvios:

1) JOR (Salto com offset por registro):

Sintaxe: JOR RfA

Operação: PC PC + RfA

Resumo: o conteúdo do registro RfA, em complemento de dois, é adicionado ao contador de programa (PC). Deste modo permite um salto de +32.768 a -32.767 instruções para frente e para trás, respectivamente, contadas a partir da instrução seguinte ao JOR.

2) JOI (Salto com offset imediato):

Sintaxe: JOI y

Operação: PC PC + y

Resumo: o valor y em complemento de dois, é adicionado ao PC. Permite saltos de +32.768 a -32.767 instruções à partir da instrução seguinte ao JOI.



3) JEI (Salto Efetivo Imediato):

Sintaxe: JEI y

Operação: PC y

Resumo: o PC recebe o endereço efetivo y para o salto.

4) JER (Salto Efetivo por Registro):

Sintaxe: JER RfA

Operação: PC RfA

Resumo: o PC recebe o endereço efetivo para o salto, contido no registro RfA.

5) JBS (Teste de bit específico para salto):

Sintaxe: JBS RfA, RfB, y

Operação: PC PC + y @ RfA # RfB = 1

Resumo: se o bit RfB do registro RfA estiver setado, o PC é somado com o valor y. Permite saltos de +32.768 a -32.767 instruções à partir da instrução seguinte ao JBS. (OBS: O símbolo '@' representa a condicional 'se', e a notação RfA # RfB significa: O bit RfB do registro RfA).

6) JBC (Teste de bit específico para salto complementar):

Sintaxe: JBC RfA, RfB, y

Operação: PC PC + y @ RfA # RfB = 0

Resumo: se o bit RfB do registro RfA estiver em zero, o PC é somado com o valor y (expresso em complemento de dois). Permite saltos de +32.768 a -32767 instruções à partir da instrução seguinte ao JBC. (OBS: O símbolo '@' representa a condicional "se", e a notação RfA # RfB significa: O bit RfB do registro RfA).



7) COR (Chamada de sub-rotina com offset por registro):

Sintaxe: COR RfA

Operação: (SP) PC; SP SP + 1; PC PC + RfA

Resumo: o conteúdo do PC é salvo na pilha e o apontador de topo da pilha (SP) é incrementado. Em seguida, o programa sofre um salto com offset dado pelo registro RfA.

8) COI (Chamada de sub-rotina com offset imediato):

Sintaxe: COI y

Operação: (SP) PC; SP SP + 1; PC PC + y

Resumo: o conteúdo do PC é salvo na pilha e o apontador de topo da pilha (SP) é incrementado. Em seguida, o programa sofre um salto com offset dado pelo valor y.

9) CEI (Chamada de sub-rotina para endereço Efetivo Imediato)

Sintaxe: CEI y

Operação: (SP) PC; SP SP + 1; PC y

Resumo: o conteúdo do PC é salvo na pilha e o apontador de topo da pilha (SP) é incrementado. Em seguida, o programa é desviado para o endereço efetivo y.

10) CER (Chamada de sub-rotina para endereço efetivo por registro):

Sintaxe: CER RfA

Operação: (SP) PC; SP SP + 1; PC RfA

Resumo: o conteúdo do PC é salvo na pilha e o apontador de topo da pilha (SP) é incrementado. Em seguida, o programa é desviado para o endereço efetivo contido no registro RfA.



11) CBS (Teste de bit específico para chamada de sub-rotina):

Sintaxe: CBS RfA,RfB,y

Operação: (SP) PC; SP SP + 1; PC Pc + y @ RfA # RfB =1

Resumo: se o bit RfB do registro RfA estiver setado, o conteúdo do PC é salvo na pilha e o apontador de topo da pilha (SP) é incrementado. Em seguida, o programa sofre um salto com offset dado pelo valor y (OBS: O símbolo '@' representa a condicional 'se', e a notação RfA # RfB significa: O bit RfB do registro RfA).

12) CBC (Teste de bit específico para chamada complementar de sub-rotina):

Sintaxe: CBC RfA,RfB,y

Operação: (SP) PC; SP SP + 1; PC Pc + y @ RfA # RfB =1

Resumo: se o bit RfB do registro RfA estiver em zero, o conteúdo do PC é salvo na pilha e o apontador de topo da pilha (SP) é incrementado. Em seguida, o programa sofre um salto com offset dado pelo valor y (expresso em complemento de 2). (OBS: O símbolo '@' representa a condicional 'se', e a notação RfA # RfB significa: O bit RfB do registro RfA).

13) RET (Retorno da sub-rotina):

Sintaxe: RET

Operação: SP SP - 1; PC (SP)

Resumo: o conteúdo do PC guardado na pilha é restaurado e o programa retorna então ao ponto em que ocorreu a chamada da sub-rotina.



c) Instruções de Operações entre Registros:

1) MOV (Movimenta Dado de Registro para Registro):

Sintaxe: MOV Rd, RfA

Operação: RD RfA

Resumo: o conteúdo de registro RfA é copiado para registro Rd.

2) ADD (Soma dos conteúdos de dois registros):

Sintaxe: Add Rd,RfA

Operação: Rd Rd + RfA

Resumo: os conteúdos dos registros Rd e RfA são somados e o resultado colocado no registro Rd.

3) ADC (Soma os conteúdos de dois registros e o carry):

Sintaxe: ADC Rd,RfA

Operação: Rd Rd + RfA + C

Resumo: os conteúdos dos registros Rd e RfA e o carry são somados e o resultado colocado no registro Rd.

4) SUB (Subtração dos conteúdos de dois registros):

Sintaxe: SUB Rd,RfA

Operação: Rd Rd + RfA

Resumo: o conteúdo do registro Rd é subtraído do conteúdo de RfA e o resultado colocado em Rd.

5) SBC (Subtração dos conteúdos de dois registros com o carry):

Sintaxe: SBC Rd,RfA

Operação: Rd Rd - RfA - C

Resumo: o conteúdo do registro Rd é subtraído do conteúdo de RfA e do carry. O resultado é colocado em Rd.



6) AND (Operação Lógica 'E'):

Sintaxe: AND Rd,RfA

Operação: Rd Rd & RfA

Resumo: a operação lógica 'E' é realizada entre os registros Rd e RfA bit a bit. O resultado é colocado em Rd.

7) OR (Operação Lógica 'OU'):

Sintaxe: OR Rd,RfA

Operação: Rd Rd ^ RfA

Resumo: a operação lógica 'OU' é realizada entre os registros Rd e RfA bit a bit. O resultado é colocado em Rd.

8) XOR (Operação Lógica 'OU-EXCLUSIVO'):

Sintaxe: XOR Rd,RfA

Operação: Rd Rd * RfA

Resumo: a operação lógica 'OU-EXCLUSIVO' é realizada entre os registros Rd e RfA bit a bit. O resultado é colocado em Rd.

9) RTR (Rotação para a Direita):

Sintaxe: RTR RfA,y

Operação: RfA > y

Resumo: o registro RfA é rotacionado para a direita y vezes.

10) RTL (Rotação para a Esquerda):

Sintaxe: RTL RfA,y

Operação: RfA < y

Resumo: o registro RfA é rotacionado para a esquerda y vezes.



11) RRC (Rotação para a Direita com o carry):

Sintaxe: RRC RfA,y

Operação: RfA,C > y

Resumo: o registro RfA juntamente com o carry é rotacionado para a direita y vezes.

12) RLC (Rotação para a Esquerda com o carry):

Sintaxe: RLC RfA,y

Operação: RfA,C < y

Resumo: o registro RfA juntamente com o carry, é rotacionado para a esquerda y vezes.

13) NOT (Operação Lógica 'NOT'):

Sintaxe: NOT Rd,RfA

Operação: RfA / RfA

Resumo: o registro RfA é complementado bit a bit.

A tabela III.1 mostra um resumo das instruções do microcontrolador com os seus códigos e resumo de operação de cada uma delas.



Tabela III.1 - Conjunto de Instruções para o Microprocessador

	Código	Mnemônico	Operando	Comentários	Tipo
1.	000000	LDM	Rd, y	$Rd \leftarrow (y)$	Transferência de Dados
2.	000001	LDR	Rd, RfB	$Rd \leftarrow ((RfB))$	
3.	000010	LDI	Rd, y	$Rd \leftarrow y$	
4.	000011	LDG	Rd, Ga	$Rd \leftarrow (Ga)$	
5.	000100	STM	Rfa, y	$(y) \leftarrow (Rfa)$	
6.	000101	STR	Rfa, RfB	$((Rfb)) \leftarrow (RfA)$	
7.	000110	STG	Rfa, Ga	$(Ga) \leftarrow (RfA)$	
8.	001000	JOR	Rf, y	$PC \leftarrow PC + Rf$	Saltos e Desvios
9.	001001	JOI	y	$PC \leftarrow PC + y$	
10.	001010	JEI	y	$PC \leftarrow y$	
11.	001011	JER	Rf	$PC \leftarrow Rf$	
12.	001100	JBS	Rfa, Rfb, y	$PC \leftarrow PC + y @ Rfa\#Rfb=1$	
13.	001101	JBC	Rfa, Rfb, y	$PC \leftarrow PC + y @ Rfa\#Rfb=0$	
14.	001110	COR	Rf	$SP \leftarrow PC; SP \leftarrow SP + 1; PC \leftarrow PC + Rf$	
15.	001111	COI	y	$SP \leftarrow PC; SP \leftarrow SP + 1; PC \leftarrow PC + y$	
16.	010000	CEI	y	$SP \leftarrow PC; SP \leftarrow SP + 1; PC \leftarrow Y$	
17.	010001	CER	Rf	$SP \leftarrow PC; SP \leftarrow SP + 1; PC \leftarrow Rf$	
18.	010010	CBS	Rfa, Rfb, y	$SP \leftarrow PC; SP \leftarrow SP + 1; PC \leftarrow PC + Y @ Rfa\#Rfb=1$	
19.	010011	CBC	Rfa, Rfb, y	$SP \leftarrow PC; SP \leftarrow SP + 1; PC \leftarrow PC + Y @ Rfa\#Rfb=0$	
20.	010100	RET	-	$SP \leftarrow SP - 1; PC \leftarrow SP$	
21.	011000	MOV	Rd, Rf	$Rd \leftarrow Rf$	Operações entre Registros
22.	011001	ADD	Rd, Rf	$Rd \leftarrow Rd + Rf$	
23.	011010	ADC	Rd, Rf	$Rd \leftarrow Rd + Rf + C$	
24.	011011	SUB	Rd, Rf	$Rd \leftarrow Rd - Rf$	
25.	011100	SBC	Rd, Rf	$Rd \leftarrow Rd - Rf - C$	
26.	011101	AND	Rd, Rf	$Rd \leftarrow Rd \& Rf$	
27.	011110	OR	Rd, Rf	$Rd \leftarrow Rd \wedge Rf$	
28.	011111	XOR	Rd, Rf	$Rd \leftarrow Rd * Rf$	
29.	100000	RTR	Rd, y	$Rd \gg y$	
30.	100001	RTL	Rd, y	$Rd \ll y$	
31.	100010	RRC	Rd, y	$Rd, C \gg y$	
32.	100011	RLC	Rd, y	$Rd, C \ll y$	
33.	100100	NOT	Rd, Rf	$Rd \leftarrow \text{NOT } Rf$	
34.	111111	HLT	-	Parada de Programa	Outras

Capítulo IV

A Arquitetura Interna do Microcontrolador

IV.1 - Introdução

A arquitetura interna do microcontrolador foi definida possuindo 3 estágios de pipeline, cada um correspondendo a uma máquina independente. Deste modo, os “pipelines” foram denominados de: máquina de busca, para o estágio de busca; máquina de execução para o estágio de execução e máquina de armazenamento, para o estágio de armazenamento. As máquinas de estados correspondentes ao controle de cada “pipeline” foram desenvolvidas, sendo apresentadas e discutidas neste capítulo.

IV.2 - Máquina de Busca

A máquina de busca tem por função retirar as instruções da memória e alojá-las em uma FIFO interna (unidade de memória que libera primeiro o dado mais antigo que foi armazenado, ou seja, o primeiro dado a entrar nesta memória é também o primeiro a sair), controlando esta FIFO e enviando as instruções para a máquina de execução.

Sempre que a máquina de busca obtiver permissão de acesso ao barramento, ela colocará uma instrução em sua FIFO. Isto fará com que a velocidade do microcontrolador aumente, otimizando o tempo de acesso à uma instrução, pois, no instante em que a máquina de execução requisitar uma instrução esta estará disponível, não sendo necessário realizar uma busca à memória. Ou seja, a máquina utiliza o tempo ocioso do barramento para trazer uma função para a FIFO.



Esta máquina é também a responsável pela decodificação das instruções de salto e retorno, pois estas instruções modificam a sequência de execução do programa. Neste caso, quando ocorrer uma instrução de salto ou retorno ela têm a responsabilidade de armazenar o endereço em que o programa estava sendo executado. A figura IV.1 mostra o diagrama de estados da máquina de busca.

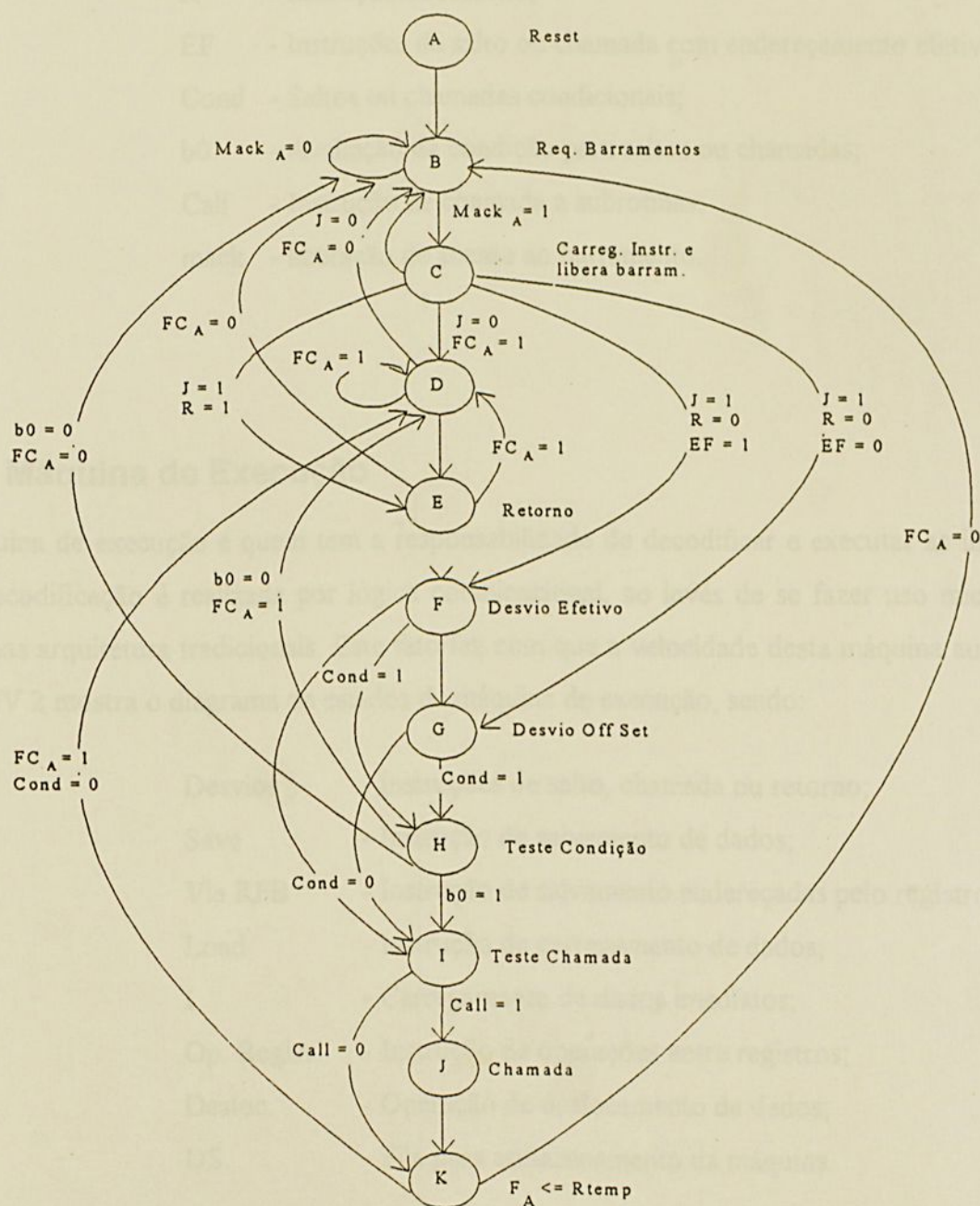


Figura IV.1 - Diagrama de Estados da Máquina de Busca

No diagrama da figura IV.1, a seguinte simbologia foi adotada:

FC _A	- FIFO da máquina A cheia;
J	- Instruções de salto ou chamadas;
R	- Instrução de retorno;
EF	- Instruções de salto ou chamada com endereçamento efetivo;
Cond	- Saltos ou chamadas condicionais;
b0	- Avaliação da condição para saltos ou chamadas;
Call	- Instrução de chamada a subrotinas;
mack	- liberação de acesso ao barramento.

IV.3 - Máquina de Execução

A máquina de execução é quem tem a responsabilidade de decodificar e executar as instruções. Esta decodificação é realizada por lógica combinacional, ao invés de se fazer uso microcódigo como nas arquitetura tradicionais. Este fato faz com que a velocidade desta máquina aumente. A figura IV.2 mostra o diagrama de estados da máquina de execução, sendo:

Desvios	- Instruções de salto, chamada ou retorno;
Save	- Instrução de salvamento de dados;
Via RFB	- Instrução de salvamento endereçadas pelo registro RFB;
Load	- Instrução de carregamento de dados;
I	- Carregamento de dados imediatos;
Op. Regist.	- Instrução de operações entre registros;
Desloc.	- Operação de deslocamento de dados;
DS.	- Fila para armazenamento da máquina.



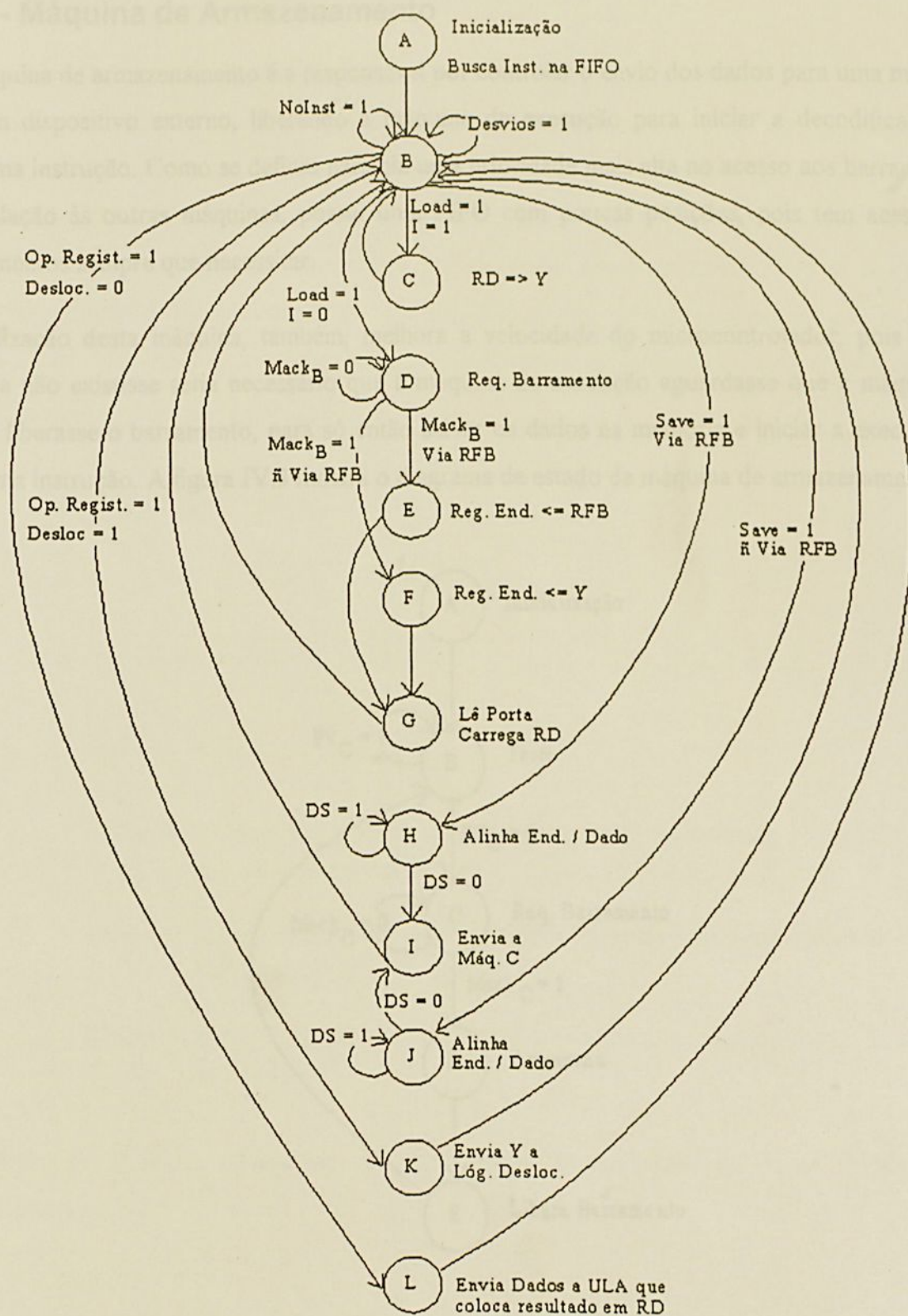


Figura IV.2 - Diagrama de Estados da Máquina de Execução

IV.4 - Máquina de Armazenamento

A máquina de armazenamento é a responsável por controlar o envio dos dados para uma memória ou um dispositivo externo, liberando a máquina de execução para iniciar a decodificação da próxima instrução. Como se definiu para ela uma prioridade mais alta no acesso aos barramentos, em relação às outras máquinas, possui uma FIFO com poucas posições, pois tem acesso aos barramentos sempre que necessitar.

A utilização desta máquina, também, melhora a velocidade do microcontrolador, pois caso a mesma não existisse seria necessário que a máquina de execução aguardasse que a máquina de busca liberasse o barramento, para só então salvar os dados na memória e iniciar a execução da próxima instrução. A figura IV.3 mostra o diagrama de estado da máquina de armazenamento.

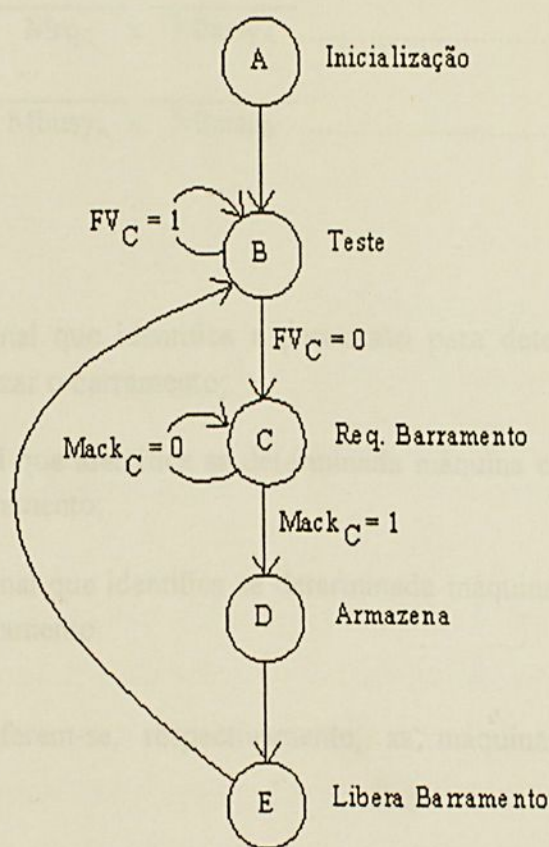


Figura IV.3 - Diagrama de Estado da Máquina de Armazenamento

IV.5 - Gerenciador de Acesso

Uma outra máquina de fundamental importância dentro da arquitetura proposta é o “Gerenciador de Acesso”. Em função das três máquinas utilizarem os barramentos de dados e endereços conjuntamente, pois trabalham em paralelo, pode acontecer que, em um determinado momento, pelo menos duas delas necessitem utilizá-los ao mesmo tempo. Por este motivo existe na arquitetura do microcontrolador um gerenciador de acesso aos barramentos. A prioridade dada por ele foi definida na seguinte sequência: 1ª máquina de armazenamento, 2ª máquina de execução e 3ª máquina de busca. As equações 1, 2 e 3 apresentam as relações lógicas entre estas três máquinas.

$$Mack = Mrq_A \times \overline{Mrq_B} \times \overline{Mrq_C} \dots\dots\dots IV.1$$

$$Mack = Mrq_B \times \overline{Mrq_C} \times \overline{Mbusy_A} \dots\dots\dots IV.2$$

$$Mack = Mrq_C \times \overline{Mbusy_A} \times \overline{Mbusy_B} \dots\dots\dots IV.3$$

Sendo:

Mack = O sinal que identifica a permissão para determinada máquina utilizar o barramento;

Mrq = O sinal que identifica se determinada máquina está requisitando o barramento;

Mbusy = O sinal que identifica se determinada máquina está utilizando o barramento.

Os índices A, B e C referem-se, respectivamente, as máquinas de busca, execução e armazenamento.



IV.6 - Arquitetura Proposta para o Microcontrolador

A figura IV.4 mostra a arquitetura elaborada do microcontrolador proposto.

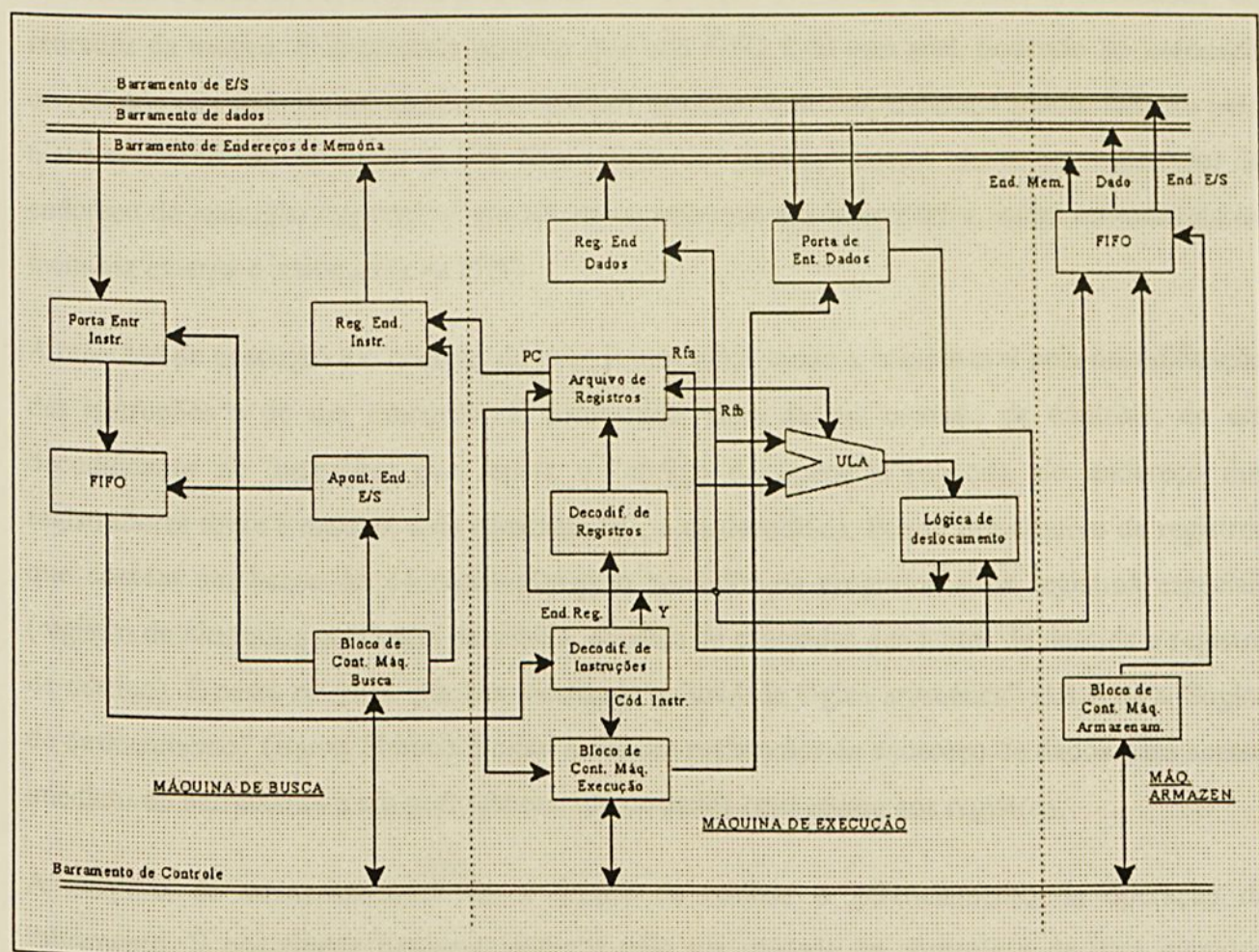


figura IV.4 - Arquitetura do microcontrolador proposto

A arquitetura proposta para o microcontrolador possui 4 barramentos internos, sendo: barramento de entrada/saída, que é o barramento que faz a ligação direta com as portas quando se deseja atuar diretamente em um pino ou em uma série deles, este barramento foi colocado em função da ação dedicada que se deseja deste microcontrolador; barramentos de dados, por onde circulam os dados recebidos e a serem armazenados na memória; barramento de endereços de memória, por onde devem ser enviados os endereços dos dados a serem recebidos ou armazenados e barramento de controle, por onde as máquinas de controle se comunicam.

A máquina de busca possui basicamente, além da sua unidade controladora, um porto de entradas que faz a interface com o mundo externo, uma memória do tipo FIFO e o apontador de endereço de entrada e saída da FIFO, para determinar a localização dos dados que serão armazenados ou retirados da mesma. É importante ressaltar que esta FIFO é do tipo anel, podendo ser colocado ou retirado um dado de seu interior de qualquer posição, com este endereço sendo definido pelo bloco apontador de endereços de E/S. Este estágio de “pipeline” possui, ainda, um registrador de endereço de instrução responsável em apontar o endereço das instruções a serem retiradas da memória externa e introduzida na FIFO.

A máquina de execução é o coração do microcontrolador sendo a responsável pela decodificação das instruções e de sua execução. É constituída, além da sua unidade de controle, de um decodificador de instruções que separa o “código de operação”(opcod), do endereço dos registradores fonte e destino, e da constante “y”, e os decodifica. Possui um arquivo de registros que contém um banco de 64 registradores, o contador de programa e os registros especiais (registro de chamadas pendentes de subida, registro de chamadas pendentes de descida e registro de destino). É nesta máquina que fica localizada a “unidade lógica aritmética” (ULA) e a “lógica de deslocamento” (Barrel Shifter). Possui ainda um registrador de endereço de dados para acesso imediato ao barramento e um porto de entrada de dados com acesso aos registros.

A máquina de armazenamento tem a função de realizar o gerenciamento da operação de salvamento. É constituída fundamentalmente pela sua unidade de controle e de uma memória FIFO.



Capítulo V

Simulação do Microcontrolador em Linguagem Verilog-HDL

V.1 - Introdução

Este capítulo apresenta a simulação, utilizando linguagens de descrição de hardware, das máquinas seqüenciais que constituem os blocos de controle dos 3 “pipelines” definidos para a arquitetura do microcontrolador. A simulação foi realizada em uma estação de trabalho “workstation” modelo Sun SPARCstation 2. A linguagem de descrição de circuitos utilizada foi a Verilog HDL e o “software” de simulação o Verilog XL, ambos da Cadence.

V.2 - A Simulação Digital

A simulação digital se constitui em uma poderosa ferramenta de trabalho para engenheiros envolvidos com projetos de sistemas eletrônicos em geral, e em uma ferramenta insubstituível, para os projetistas de circuitos integrados em particular, pela economia de tempo e de dinheiro proporcionado pelo seu uso.

Existem vários programas que viabilizam a simulação de circuitos eletrônicos. Programas que simulam desde circuitos analógicos, circuitos digitais e mistos (analógicos e digitais juntos), fazendo a análise de tempo e do caminho crítico dos sinais a partir de um diagrama esquemático, até aqueles que realizam simulações a partir de uma descrição comportamental destes circuitos. Ou seja, hoje é possível verificar o desempenho de quase todos circuitos eletrônicos que se deseje utilizando quase que unicamente simuladores. Consegue-se, assim, obter sofisticadas análises,



ainda na concepção destes circuitos, economizando tempo e dinheiro e otimizando o desempenho dos mesmos.

Neste trabalho, até a fase que se encontra aqui relatado, fez-se a opção pelo uso de um simulador que utilizasse uma linguagem de descrição de "hardware. Neste caso, a opção recaiu sobre o programa Verilog-XL, da CADENCE, que utiliza a linguagem de programação Verilog-HDL, ao invés da linguagem VHDL, pelo apoio fornecido pela representante no Brasil da CADENCE. Este programa, assim como o VHDL, possui reconhecimento internacional e transmite a tranquilidade necessária para o desenvolvimento do trabalho. Além do mais, por ele ser hierárquico possibilita a simulação de todo o microcontrolador. Como faz parte do pacote da CADENCE, para projeto de circuitos integrados, a geração do esquemático do circuito e, futuramente, do "layout" do mesmo, é apenas uma questão de se acionar os programas corretos para que estes sejam feitos automaticamente.

Neste trabalho, foram simuladas todos os circuitos de controle dos "pipelines", sendo apresentadas as suas simulações e discutidos os seus resultados. Nesta linguagem, a sintaxe utilizada para definir uma porta qualquer, por exemplo, uma porta "e" (and), sempre obedece a seguinte sequência:

AND A1 (S, E1, E2,... ,En);

Sendo:

AND - Relação lógica entre os sinais de entrada e saída;

A1 - Nome dado a esta porta;

S - Saída desta porta;

E1 - Entrada número 1;

E2 - Entrada número 2;

En - Entrada número n.

Um exemplo de como é feita a descrição de um circuito, a partir de seu diagrama de estados, é apresentado item IV.4

333



V.3 - Circuito Preliminar - “Flip-Flop” Tipo D

O “software” Verilog-XL, que foi utilizado neste trabalho, tem disponível em sua biblioteca apenas as portas lógicas básicas para descrição de circuitos, portas como “e” (and), “ou” (or), “inversora” (not), “não e” (nand), entre outras, porém não conta com “Flip-flops” ou blocos mais complexos de circuitos digitais, tais como: contadores, registradores de deslocamento, etc... Deste modo, para a descrição dos circuitos de controle das três máquinas do microcontrolador foi necessário a implementação de “Flip-flops” a partir das portas básicas “não e”(nand) e “inversora” (not). Assim, implementou-se um “Flip-Flop” tipo “D”, ativo na borda de descida do “clock”, com “preset” e “clear” ativos em nível baixo. Um arquivo de estímulos foi criado para que estes pudessem ser aplicados as entradas deste circuito. Como resultado da simulação obteve-se as formas de onda apresentadas na figura V.1. O anexo 1 apresenta o programa de simulação desenvolvido em linguagem Verilog-HDL. Os seguintes termos foram adotados para efeito de simulação:

dt	- Entrada de dados;
clk	- Entrada do pulso de clock;
clr	- Entrada clear;
pr	- Entrada preset;
q	- Saída.

Simulação do “Flip-Flop” e Discussão dos Resultados

A figura V.1, apresenta as curvas referentes à simulação realizada do “flip-flop tipo D”. Este “flip-flop” será o utilizado na implementação das máquinas de estado do microcontrolador. Pode-se verificar, nas curvas apresentadas, que entre os instantes 0 e 30, quando o “preset” e o “clear” estão em nível alto e com isto não estão atuando nas saídas do “flip-flop” tipo D, que este responde aos sinais aplicados a sua entrada durante a descida do pulso de “clock”, ou seja, os sinais que estão aplicados a entrada “dt”, são transferidos para a saída “q”.

No intervalo entre os instantes 30 e 50, o sinal “pr” é levado para zero, fazendo com que a saída “q” assuma o nível alto, independentemente do sinal aplicado a entrada do “flip-flop”. Pode-se notar que a entrada de “preset” é assíncrona, ou seja, a saída “q” assumirá valor 1, sem



necessidade de esperar pela descida do "clock", tendo como limitação apenas o tempo de atraso das portas.

No intervalo entre os instantes 50 e 70, o "flip-flop" recebe o sinal de "clear" em nível zero, levando a saída "q" para zero. As considerações feitas para a entrada de "preset" são válidas, também, para a entrada "clear", como pode ser visto na figura V.1.

A partir do instante 70, o "flip-flop" volta a ter os comandos "preset" e "clear" em nível alto e apresenta na saída "q" na descida do pulso do "clock", o sinal que estiver aplicado na entrada "dt". Esta simulação mostra que o "flip-flop tipo D" implementado atuou como se esperava inicialmente.

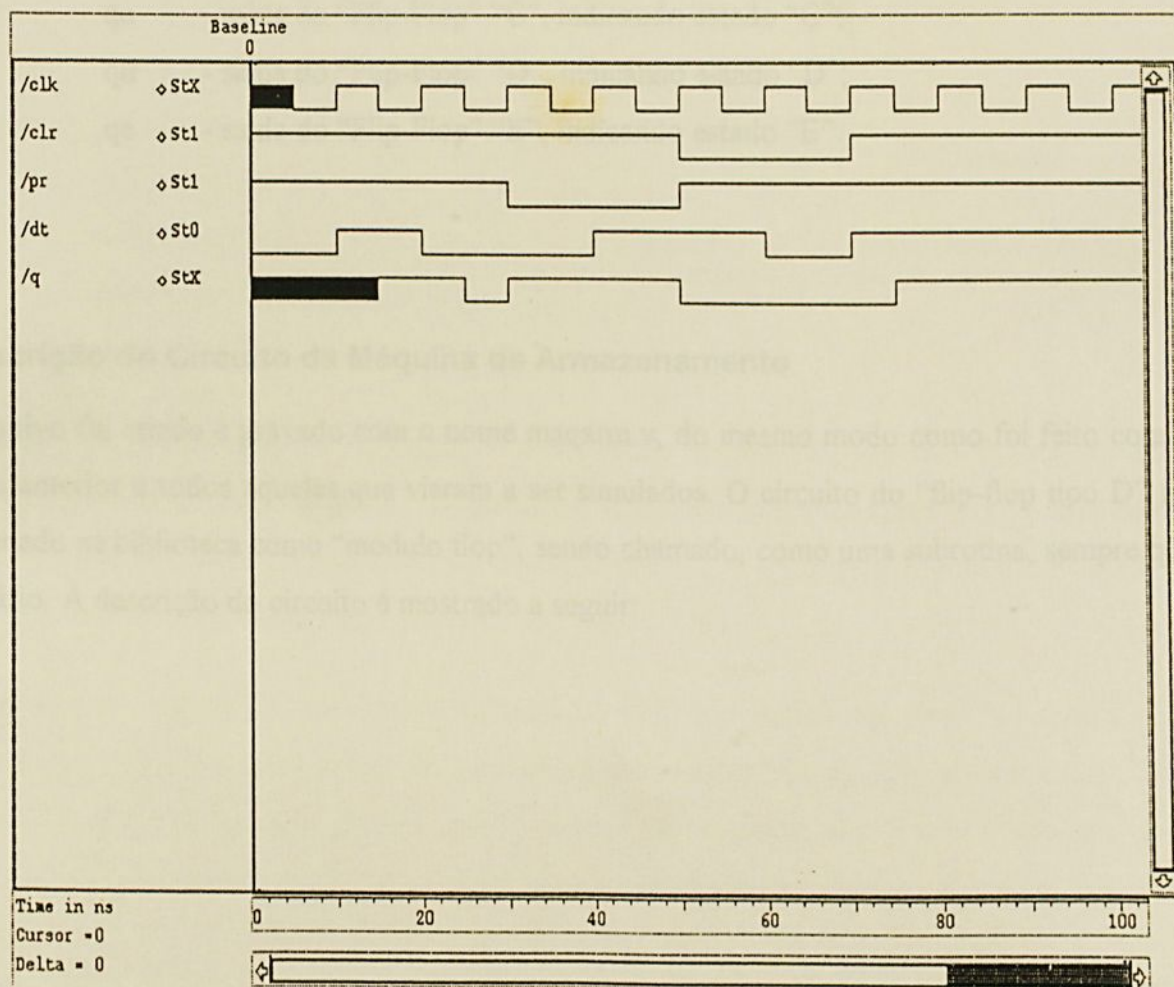


Figura V.1 - Resultados obtidos na simulação do "Flip-flop"

V.4 - Máquina de Armazenamento

A simulação da máquina de estados do segundo “pipeline”, que vem a se constituir o bloco de controle da máquina de armazenamento, foi a escolhida e utilizada para apresentar o mecanismo de descrição do circuito em linguagem Verilog-HDL. Deste modo, a partir do diagrama de estado desta máquina, apresentado na figura V.2, obteve-se a descrição necessária para se realizar a simulação desejada. Os seguintes termos foram adotados:

- mackc - Permissão de acesso aos barramentos;
- fvc - “FIFO” da máquina C vazia;
- qa - saída do “Flip-Flop” “A”, indicando estado “A”;
- qb - saída do “Flip-Flop” “B”, indicando estado “B”;
- qc - saída do “Flip-Flop” “C”, indicando estado “C”;
- qd - saída do “Flip-Flop” “D”, indicando estado “D”;
- qe - saída do “Flip-Flop” “E”, indicando estado “E”.

a) Descrição do Circuito da Máquina de Armazenamento

Um arquivo foi criado e gravado com o nome *maqarm.v*, do mesmo modo como foi feito com o circuito anterior e todos aqueles que vieram a ser simulados. O circuito do “flip-flop tipo D” foi armazenado na biblioteca como “módulo flop”, sendo chamado, como uma subrotina, sempre que necessário. A descrição do circuito é mostrado a seguir:




```
module maqarm (da, clk, pr, clr, qa, qb, qc, qd, qe, fvc, mackc);
```

```
input da, clk, pr, clr, fvc, mackc;
```

```
output qa, qb, qc, qd, qe;
```

```
flop fa (da, clk, pr, clr, qa),
```

```
fb (db, clk, clr, pr, qb),
```

```
fc (dc, clk, clr, pr, qc),
```

```
fd (dd, clk, clr, pr, qd),
```

```
fe (de, clk, clr, pr, qe),
```

```
and a1 (a, nfvc, qb),
```

```
a2 (b, nmackc, qc),
```

```
a3 (dd, mackc, qc),
```

```
a4 (de, qd),
```

```
a5 (c, fvc, qb);
```

```
or o1 (db, qa, c, qe),
```

```
o2 (dc, a, b),
```

```
not n1 (nfvc, fvc),
```

```
n2 (nmackc, mackc);
```

```
endmodule
```

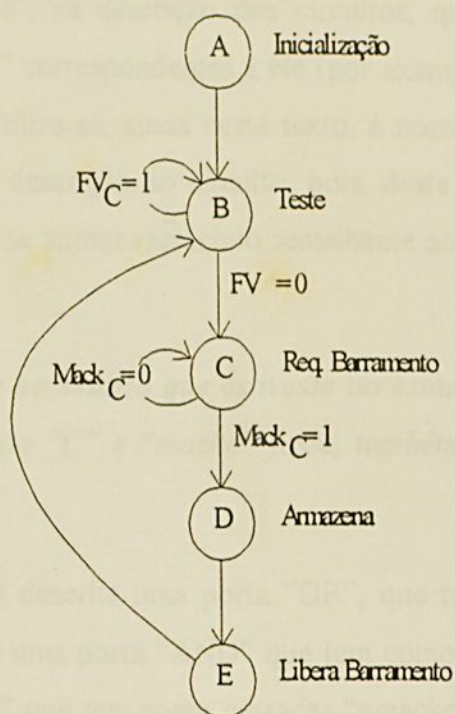


Figura V.2 - Diagrama de Estados da Máquina de Armazenamento

A descrição de um circuito na linguagem Verilog-HDL, a partir de um fluxograma, se inicia dando-se um nome ao circuito que se vai simular e indicando todos os sinais que se comunicam com ele. Define-se, então, quais serão os sinais de entrada e saída para o circuito. Finalmente define-se as relações lógicas entre os sinais de entrada e saída.

Em relação aos estados da máquina de armazenamento, que podem ser vistos no fluxograma, utilizou-se um “flip-flop” para definir cada um deles, sendo estes “flip-flops” chamados como uma subrotina, a partir do circuito gerado anteriormente. Deste modo, este módulo utilizou cinco “flip-flops”. O seguinte raciocínio foi utilizado para a elaboração da descrição do circuito:

Para a máquina de armazenamento passar ao estado “B”, seria necessário que estivesse no estado “A” ou, estivesse no estado “B” e “fvc” fosse igual a um, ou ainda que estivesse no estado “E”.

Por este motivo foi descrita uma porta "OR" que tem como saída "db" e entradas "qa", "c" e "qe", sendo "c" a saída de uma porta "AND" que tem como entradas "qb" e "fvc". Como os estados são representados pelas saídas de "flip-flops", na descrição dos circuitos, quando se referiu a um estado, utilizou-se a saída dos "flip-flops" correspondentes à ele (por exemplo: "qa" para o estado "A", "qb" para o estado "B", etc.... Utiliza-se, ainda neste texto, a nomenclatura das portas em inglês para facilitar a compreensão da descrição do circuito, pois, deste modo, a nomenclaturas ficarão semelhantes. Novamente, pode-se adotar raciocínio semelhante ao adotado anteriormente para o estado "C":

Para a máquina passar ao estado "C", seria necessário que estivesse no estado "B" e "fvc" fosse igual a zero ou estivesse no estado "C" e "mackc" fosse, também, igual a zero.

Para atender as necessidades do raciocínio acima foi descrita uma porta "OR", que tem como saída "dc" e entradas "a" e "b". Sendo "a" a saída de uma porta "AND" que tem como entradas "nfvc" e "qb"; sendo "b" a saída de uma porta "AND" que tem como entradas "nmackc" e "qc", sendo "nfvc" e "nmackc" as saídas das portas inversoras que tem como entradas, respectivamente, "fvc" e "mackc". Para o estado D:

Para se passar ao estado "D", seria necessário que estivesse no estado "C" e "mackc" fosse igual a um.

Portanto foi definida uma porta "AND" que tem como saída "dd" e entradas "qc" e "mackc". Novamente:

Para passar ao estado "E", seria necessário, apenas, que estivesse no estado "D".

Para este caso foi definida uma porta "AND" que tem como saída "de" e entrada "qd".

Os arquivos de descrição e estímulos são mostrados no anexo 1, sendo que este procedimento foi utilizado para descrever todas as máquinas de estado do microcontrolador.



b) Simulação do Circuito da Máquina de Armazenamento e Discussão dos Resultados

A figura V.3, apresenta as curvas referentes à simulação do circuito de controle da máquina de armazenamento.

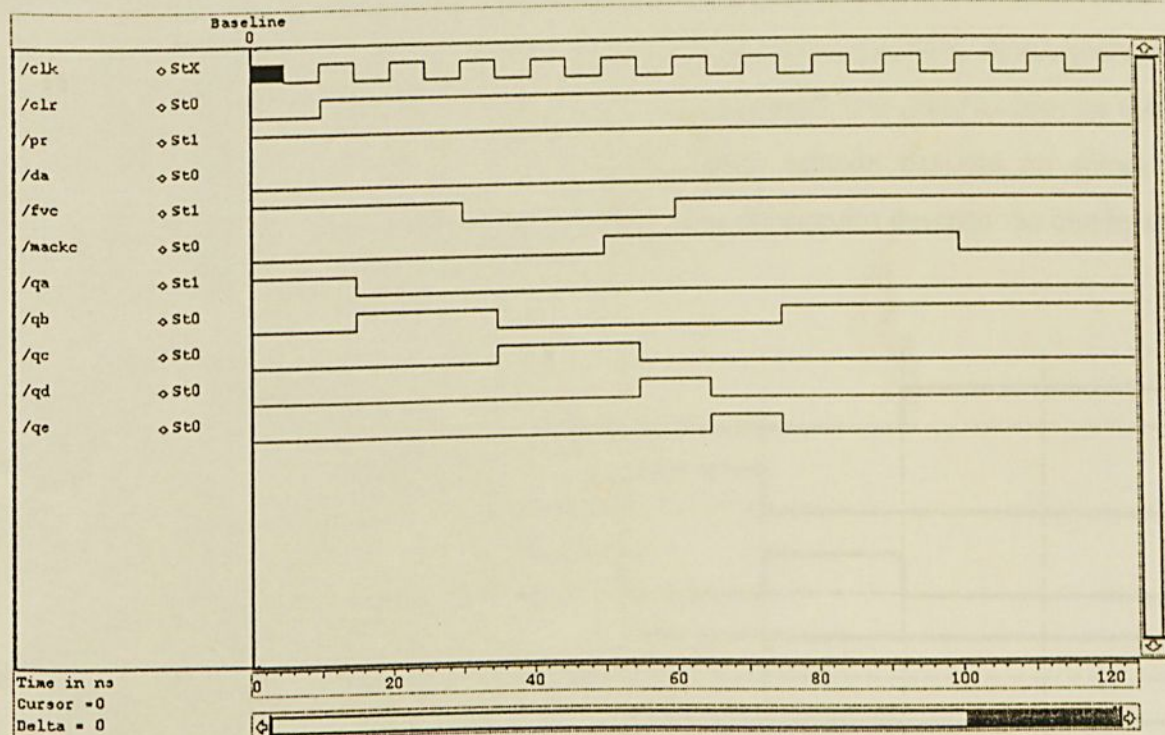


Figura V.3 - Resultados obtidos na simulação da máquina de armazenamento

Pode-se verificar que o estado "A" só é acessado no instante inicial, no momento em que se inicializa o circuito.

No instante 15, a máquina de armazenamento passa ao estado "B", onde faz o teste de FIFO para verificar a existência de dados armazenados nela. Permanece neste estado, pois "fvc" sendo igual a um (1), representa que a "FIFO" está vazia. Quando a FIFO receber um dado e o armazenar, "fvc" desce para zero e a máquina passa para o estado "C" (instante 35), requisitando o barramento, pois teria dados para serem enviados. Permanece, ainda, no estado "C", pois não possui permissão de acesso ao barramento, pois "mackc" é igual a zero. Quando o barramento é liberado passa ao estados "D"(instante 55), armazenando os dados em um dispositivo de memória (o que pode ser visto na figura V.3 quando "fvd" volta a "1", o que representa a não existência

mais de dados na FIFO). A seguir segue para o estado “E” liberando o barramento e retornando ao estado “B” (instante 75).

V.5 - Gerenciador de Acesso aos Barramentos

A descrição do circuito na linguagem Verilog-HDL pode ser feita, também, diretamente a partir de suas equações booleanas e, a partir disto, realizar a simulação. Um arquivo com os estímulos que foram aplicados as entradas deste circuito foi criado, estando descrito no anexo 1. Os resultados apresentados na figura V.4 foram obtidos a partir do circuito descrito no citado anexo.

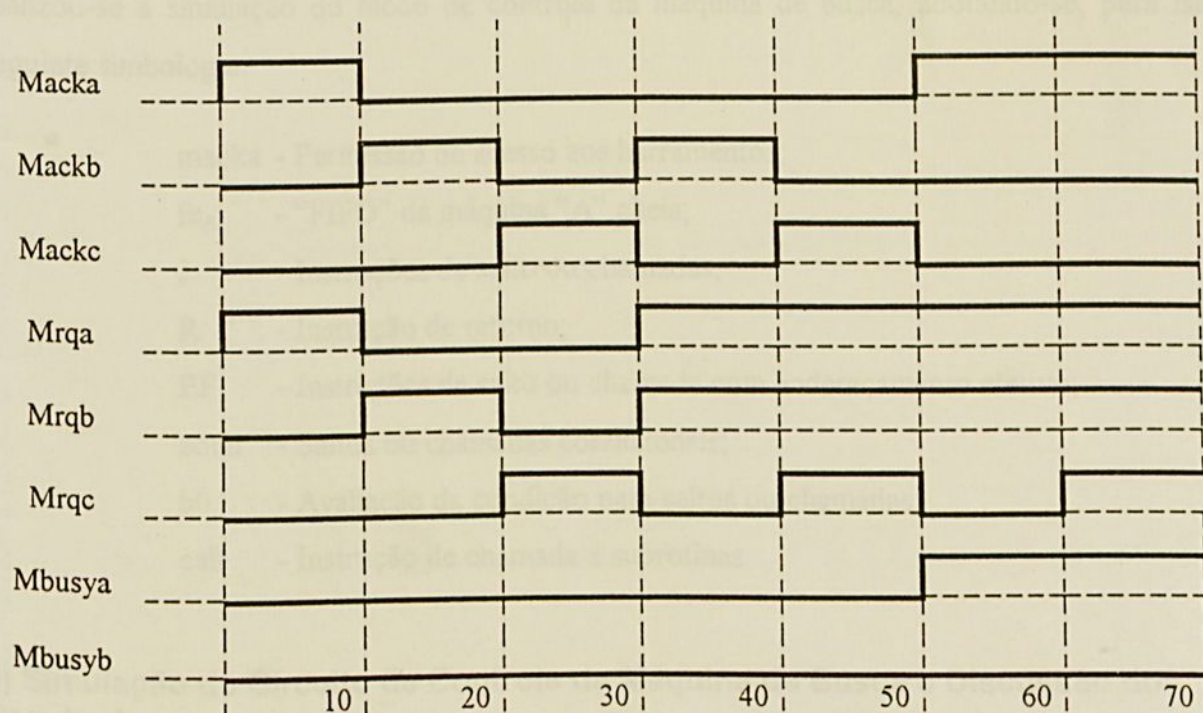


Figura V.4 - Resultados obtidos na Simulação do Gerenciador

Conforme observa-se na figura V.4, no instante zero, somente a máquina de busca requisita o barramento, recebendo, portanto, esta máquina permissão para acessá-lo. Do mesmo modo, no instante 100, somente a máquina de execução requisita o barramento, recebendo, também, permissão para acessá-lo. No instante 200 a máquina de armazenamento requisita o barramento e as demais máquinas não, portanto, esta máquina recebe permissão para acessá-lo.

No instante 400 as três máquinas requisitam o barramento simultaneamente. A máquina de armazenamento recebe permissão para acessá-lo, pois possui prioridade mais alta. No instante 500 a máquina de execução requisita o barramento, mas a máquina de busca o está utilizando, com isto, a máquina de busca mantém a permissão recebida de acessá-lo. No instante 600 as máquinas de execução e armazenamento requisitam o barramento, mas a máquina de busca continua a utilizá-lo, deste modo a máquina de busca prossegue fazendo uso do mesmo.

V.6 - Máquina de Busca

A partir de seu fluxograma, apresentado na figura IV.1, obteve-se a descrição do circuito e realizou-se a simulação do bloco de controle da máquina de busca, adotando-se, para isto, a seguinte simbologia:

- macka - Permissão de acesso aos barramentos;
- fc_A - “FIFO” da máquina “A” cheia;
- J - Instruções de salto ou chamadas;
- R - Instrução de retorno;
- EF - Instruções de salto ou chamada com endereçamento efetivo;
- cond - Saltos ou chamadas condicionais;
- b0 - Avaliação da condição para saltos ou chamadas;
- call - Instrução de chamada a subrotinas.

a) Simulação do Circuito de Controle da Máquina de Busca e Discussão dos Resultados

A figura V.5 descreve os resultados obtidos na simulação do circuito da máquina de busca. A descrição do seu circuito se encontra no anexo 1.

Como na máquina anterior, o estado “A” somente é acessado no momento da inicialização.



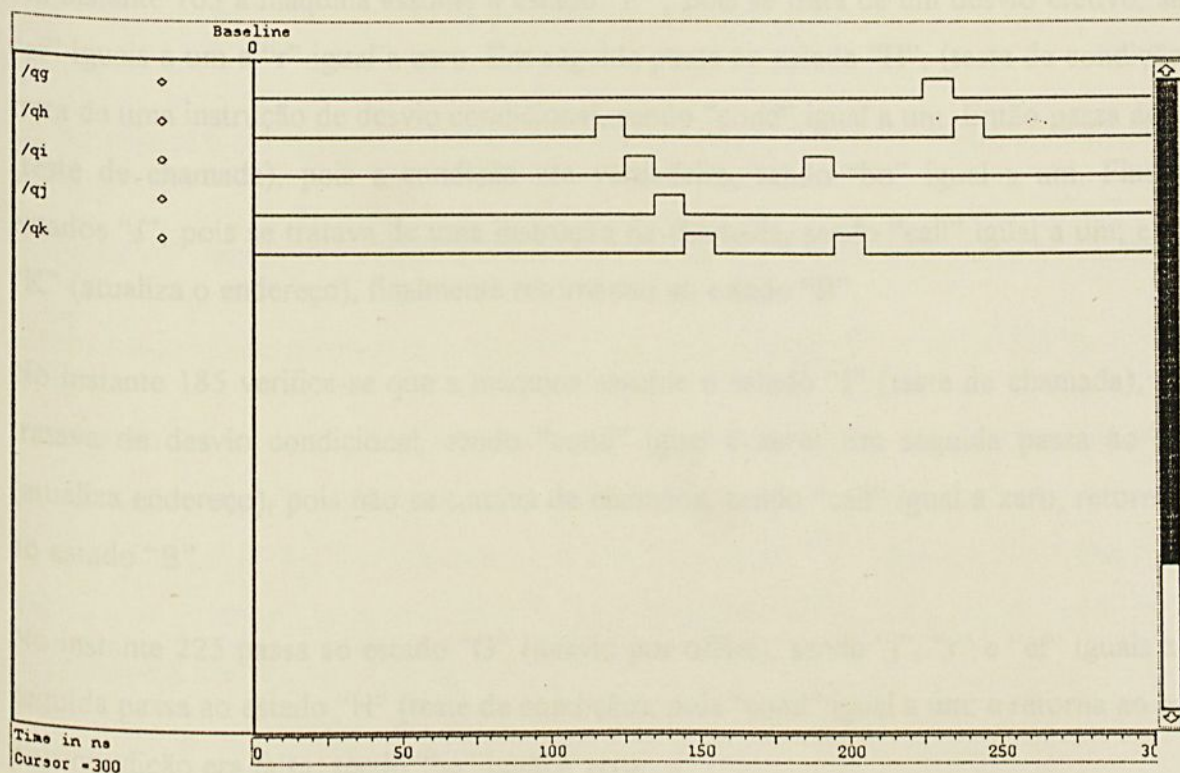
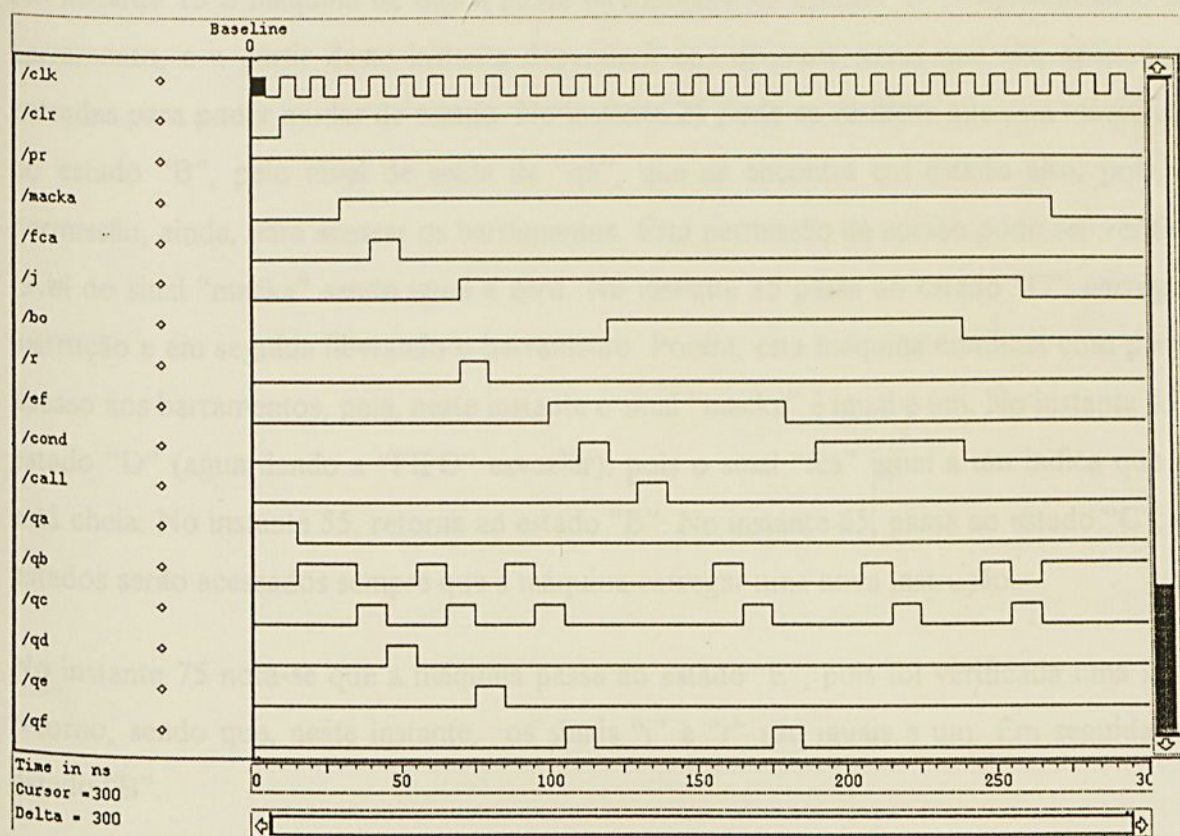


Figura V.5 - Resultados obtidos na simulação da máquina de busca

No instante 15 a máquina de busca passa diretamente ao estado “B”, requisitando o acesso ao barramento, e a partir deste instante dependerá dos diversos sinais que são aplicados as suas entradas para poder mudar de estado. No instante 25 pode-se verificar que esta máquina continua no estado “B”, pelo nível de saída de “qb”, que se encontra em estado alto, pois, não teve permissão, ainda, para acessar os barramentos. Esta permissão de acesso pode ser verificado pelo nível do sinal “macka” sendo igual a zero. No instante 35 passa ao estado “C”, carregando uma instrução e em seguida liberando o barramento. Porém, esta máquina continua com permissão de acesso aos barramentos, pois, neste instante o sinal “macka” é igual a um. No instante 45 passa ao estado “D” (aguardando a “FIFO” esvaziar), pois o sinal “fca” igual a um indica que a “FIFO” está cheia. No instante 55, retorna ao estado “B”. No instante 65, passa ao estado “C”, estes dois estados serão acessados sempre que a máquina carregar uma nova instrução.

No instante 75 nota-se que a máquina passa ao estado ‘E’, pois foi verificada uma instrução de retorno, sendo que, neste instante, os sinais “j” e “r” são iguais a um. Em seguida retorna ao estado “B”.

No instante 105 a máquina assume o estado “F”, pois se trata de um desvio efetivo, sendo “j” e “ef” iguais a um e “r” igual a zero. Em seguida passa ao estado “H”, (teste de condição), pois se trata de uma instrução de desvio condicional, sendo “cond” igual a um. Então passa ao estado “I” (teste de chamada), pois a condição era verdadeira, sendo “bo” igual a um. Finalmente aos estados “J”, pois se tratava de uma instrução de chamada, sendo “call” igual a um; e a instrução “K” (atualiza o endereço), finalmente retornando ao estado “B”.

No instante 185 verifica-se que a máquina assume o estado “I” (teste de chamada), pois não se tratava de desvio condicional, sendo “cond” igual a zero. Em seguida passa ao estado “K” (atualiza endereço), pois não se tratava de chamada, sendo “call” igual a zero, retornando, após, ao estado “B”.

No instante 225 passa ao estado “G” (desvio por offset), sendo “j”, “r” e “ef” iguais a zero. Em seguida passa ao estado “H” (teste de condição), pois “cond” igual a um; e retorna ao estado “B”, pois condição era falsa, sendo “bo” igual a zero.



No instante 265 retorna ao estado “B”, pois não era uma instrução de salto, chamada ou retorno, sendo “j” igual a zero. No instante 275 permanece no estado “B”, pois não tem permissão de acesso ao barramento, sendo “macka” igual a zero.

V.7 - Máquina de Execução

A partir de seu diagrama de estado, descreve-se seu circuito e realiza-se sua simulação. Sendo:

mckb	- Permissão de acesso aos barramentos;
dsvios	- Instruções de salto, chamada ou retorno;
sve	- Instrução de salvamento de dados;
va RFB	- Instrução de salvamento endereçadas pelo registro RFB;
lad	- Instrução de carregamento de dados;
I	- Carregamento de dados imediatos;
o. Regist.	- Instrução de operações entre registros;
dsloc.	- Operação de deslocamento de dados;
ds	- Fila para armazenamento da máquina C.

a) Simulação do Circuito de Controle da Máquina de Execução e Discussão dos Resultados

A figura V.6 mostra os resultados obtidos na simulação do circuito da máquina de execução.

Como nas máquinas anteriores, o estado “A” somente é acessado no momento da inicialização.

No instante 15 a máquina passa ao estado “B”, e só sairá deste quando tiver instruções a serem executadas e esta instrução não for uma instrução de desvio, sendo respectivamente “noins” ou “desvios” igual a um. No instante 45, passa ao estado “C” (carrega dado imediatamente no registrador de dado), pois se tratava de um carregamento imediato, sendo “load” e “i” iguais a um. Em seguida retorna ao estado “B”.



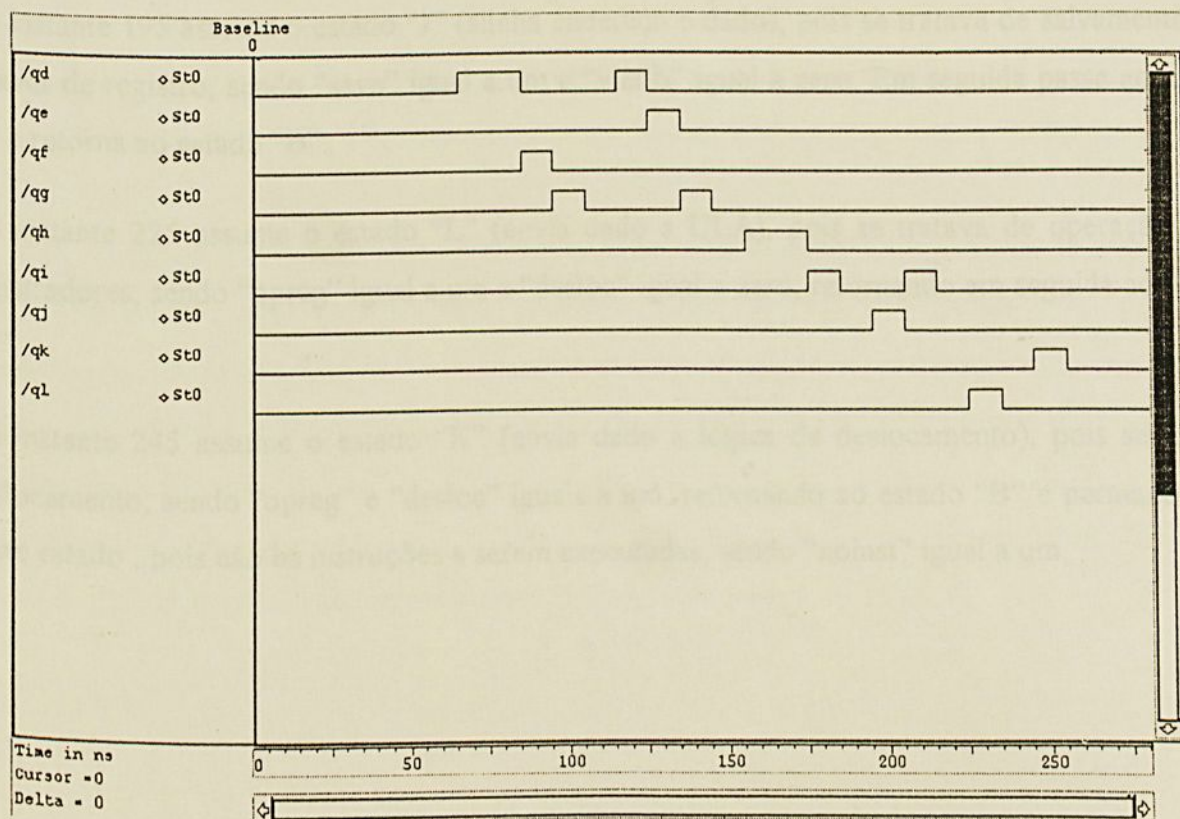
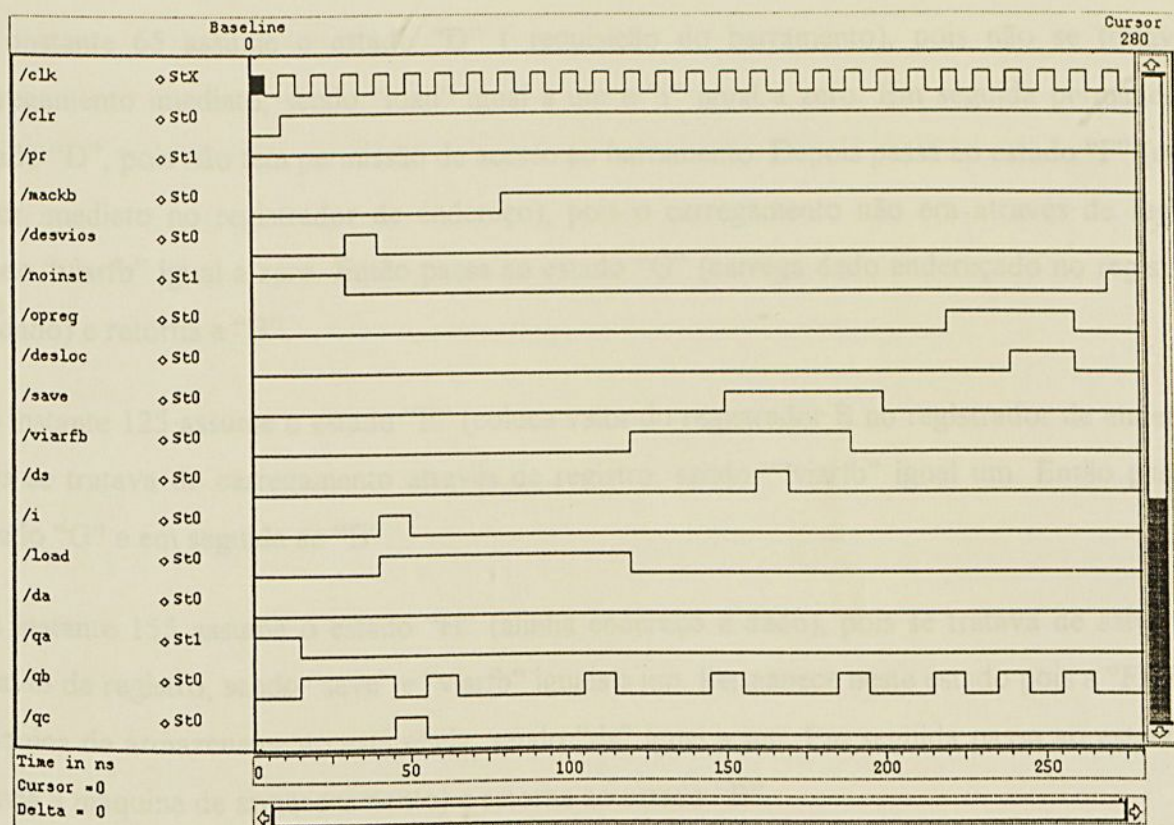


Figura V.6 - Resultados obtidos na simulação da máquina de execução

No instante 65 assume o estado “D” (requisição do barramento), pois não se tratava de carregamento imediato, sendo “load” igual a um e “i” igual a zero. Em seguida permanece no estado “D”, pois não tem permissão de acesso ao barramento. Depois passa ao estado “F” (coloca valor imediato no registrador de endereço), pois o carregamento não era através de registro, sendo “viarfb” igual a zero. Então passa ao estado “G” (carrega dado endereçado no registrador de dado) e retorna a “B”.

No instante 125 assume o estado “E” (coloca valor do registrador B no registrador de endereço), pois se tratava de carregamento através de registro, sendo “viarfb” igual um. Então passa ao estado “G” e em seguida ao “B”.

No instante 155 assume o estado “H” (alinha endereço e dado), pois se tratava de salvamento através de registro, sendo “save” e “viarfb” iguais a um. Permanece neste estado pois a “FIFO” da máquina de armazenamento está cheia, sendo “ds” igual a um. Em seguida passa ao estado “qi” (envia a máquina de armazenamento) e retorna ao estado “B”.

No instante 195 assume o estado “J” (alinha endereço e dado), pois se tratava de salvamento não através de registro, sendo “save” igual a um e “viarfb” igual a zero. Em seguida passa ao estado “I” e retorna ao estado “B”.

No instante 225 assume o estado “L” (envia dado a ULA), pois se tratava de operação entre registradores, sendo “opreg” igual a um e “desloc” igual a zero, retornando em seguida ao estado “B”.

No instante 245 assume o estado “K” (envia dado a lógica de deslocamento), pois se tratava deslocamento, sendo “opreg” e “desloc” iguais a um, retornando ao estado “B” e permanecendo neste estado, pois não há instruções a serem executadas, sendo “noinst” igual a um.



V.8 Circuito de Controle

a) Seqüência de instruções simuladas pelo circuito de controle

A intenção de se simular o circuito de controle, além das máquinas seqüenciais de cada “pipeline”, é de se poder verificar a comunicação entre elas. Para simulação deste módulo tomou-se como base, para verificação do circuito desenvolvido, as instruções descritas abaixo:

LDM

LDI

ADD

STR

RTR

JOR

Inicialmente, realizou-se simulação destas instruções, como um modo de verificar o estado de funcionamento da interligação e da comunicação existente entre os circuitos de controle das 3 máquinas. Acredita-se, que, embora um maior número de instruções necessitem ser simuladas, a perfeita execução das que aqui estão apresentadas permite dar prosseguimento ao projeto com um maior grau de confiança. Durante o desenrolar do mesmo, e com a simulação de outra partes de sua arquitetura, as outras instruções serão verificadas de modo a convalidar o circuito como um todo. A figura V.7 apresenta os resultados obtidos na simulação do circuito de controle do microcontrolador. Como notação para os estados das máquinas seqüenciais dos “pipelines” adotou-se os seguintes critérios: “Aa” significa estado “A” do bloco controlador da máquina de busca; “Ba”, estado “B” deste mesmo bloco. “Ab” significa estado “A” do bloco controlador da máquina de execução e “Ac” significa estado “A” do bloco controlador da máquina de armazenamento. Ou seja, as letras “a”, “b” e “c” indicam o bloco do circuito controlador das máquinas de busca, execução e armazenamento, respectivamente.



b) Discussão dos Resultados Obtidos na Simulação e Discussão dos Mesmos

Pode-se verificar que os estados “Aa”, “Ab” e “Ac” somente são acessados quando se inicializa a máquina, como pode ser visto entre os instantes 0 e 15, ou quando se “reinicializa” a máquina, como pode ser visto no instante 210. No instante 15 a máquina de busca passa ao estado “Ab”, a máquina de execução ao estado “Ab” e a máquina de armazenamento ao estado “Ac”.

No instante 25 a máquina de busca passa ao estado “Ca”, pois tem permissão de acesso aos barramentos, carrega a instrução LDM, libera o barramento e retorna a “Ba” no instante 35. Durante este tempo a máquina de execução permanece no estado “Bb”, pois não havia instrução a ser executada, estando “noinst” igual a um. A máquina de armazenamento permanece no estado “Bc, pois sua “FIFO” está vazia, sendo “fvc igual a um. No instante 35 a máquina de busca retorna ao estado “Ba”, a máquina de execução passa ao estado “Db”, pois agora existe a instrução LDM a ser executada, sendo os sinais “noisnt” e “i” iguais a zero e “load” igual a um. A máquina de armazenamento ainda permanece no estado “Bc”, aguardando a chegada de algum dado para enviar aos dispositivos externos. No instante 45 a máquina de busca permanece no estado “Ba”, pois a máquina de execução solicitou os barramentos e o recebeu pois esta possui prioridade mais elevada. A máquina de execução passa ao estado “Fb”, continuando a executar a instrução LDM. No instante 55 a máquina de execução passa ao estado “Gb”, terminando a execução da instrução LDM. Verifica-se que entre os instantes 35 e 75, a máquina de busca permaneceu no estado “Ba”, pois a máquina de execução utilizou os barramentos.

No instante 75 a máquina de busca passa ao estado “Ca”, carrega a instrução LDI e retorna ao estado “Ba”. No instante 85 a máquina de execução passa ao estado “Cb”, executando a instrução LDI e no instante 95 a máquina de busca carrega a instrução ADD.

No instante 105 a máquina de execução passa ao estado “Lb”, executando a instrução ADD.

No instante 115 a máquina de busca carrega a instrução STR e no instante 125 a máquina de execução passa ao estado “Hb” e em seguida ao estado “Ib”, executando a instrução STR.



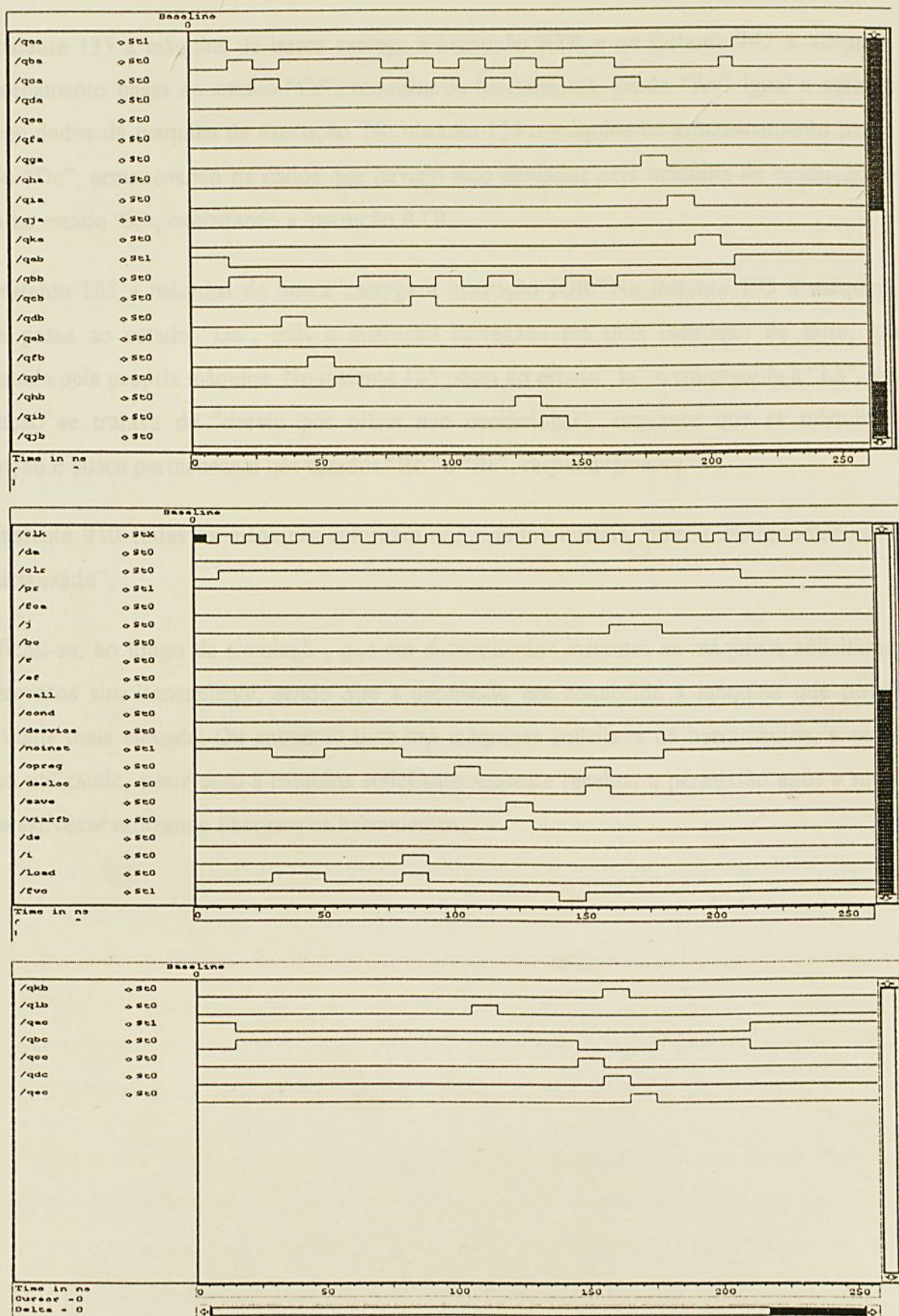


Figura V.7 - Resultados obtidos na simulação do circuito de controle



No instante 135 a máquina de busca carrega a instrução RTR e no instante 145 a máquina de armazenamento passa ao estado “Cc”, requisita os barramentos, sendo “fvc” igual a zero, pois recebeu dados da máquina de execução. No instante 155 a máquina de armazenamento passa ao estado “Dc”, armazenando os dados que haviam sido enviados pela máquina de execução. Esta passa ao estado “Kb, executando a instrução RTR.

No instante 165 a máquina de busca carrega a instrução JOR.”No instante 175 a máquina de busca passa ao estado “Ga”, pois a instrução carregada era uma instrução de salto, que é executada pela própria máquina. No instante 185 passa ao estado “Ia” e em seguida a “La”, pois a instrução se tratava de “desvio por offset não condicional”, enquanto que as máquinas de execução e busca permanecem nos estados “Bb” e “Bc”, respectivamente.

No instante 210 todas as máquinas retornam aos estados iniciais, pois o microcontrolador foi “reinicializado”.

Verificou-se, ao longo da simulação, que em determinados instantes as máquinas solicitaram os barramentos simultaneamente, sendo que a permissão era concedida à máquina que possuísse prioridade mais elevada. Ou enquanto uma das máquinas solicitava os barramentos, a outra os estava utilizando, neste caso a máquina solicitante somente recebeu a permissão após a máquina que o estivesse utilizando liberasse os barramentos.



Capítulo VI

Implementação do Circuito Desenvolvido

VI.1 - Considerações Iniciais

A implementação do circuito desenvolvido, até a presente data, será feita utilizando as ferramentas para projetos de circuitos integrados do “software” desenvolvido e comercializado pela CADENCE. Estas ferramentas permitem, sem maiores dificuldades, a geração do esquemático do circuito a ser implementado, a análise dos tempos críticos, a extração dos parâmetros exigidos para uma eventual simulação elétrica, e a geração automática de seu “layout” de integração.

Para que esta implementação possa ser feita é necessário, antes de tudo, definir a tecnologia a ser utilizada, o local de fabricação (“foundry”) do dispositivo, e a estratégia de integração (“Gate Array”, “Standard Cell”, “Full Custom”, dentre outras), pois parâmetros como estes tem influência direta sobre a confecção do “layout” de integração. No caso do dispositivo microcontrolador optou-se pelo uso da metodologia “Standard Cell”, para a sua integração, em função, principalmente, de ser um ASIP, ou seja, um circuito dedicado que, em princípio, não terá uma demanda tão grande que exija um projeto “full custom”.

A definição da tecnologia partiu da análise dos métodos disponíveis, no meio universitário, para a confecção e obtenção de um circuito integrado. Hoje, existem dois meios de se conseguir integrar um dispositivo dentro da UNESP: o primeiro utilizando o Projeto Multiusuário (PMU), coordenado pelo Centro de Tecnologia para Informática (CTI) em Campinas; e o segundo utilizando o PMU Paulista, financiado pela Fundação de Amparo à Pesquisa do Estado de



S.Paulo - FAPESP . A “foundry” escolhida, que significou ter de se optar por uma das tecnologias por ela oferecida, foi a ES2 (European Silicon Structure), sediada na França, que pode vir a ser acessada por qualquer um dos dois meios aqui apresentados (PMU ou PMU Paulista), aumentando, assim, a flexibilidade, tanto de datas, como de meios de acesso. Dentre as tecnologias existentes nesta “foundry” optou-se pela CMOS de 0,7 μm , em função desta tecnologia permitir circuitos bastante compactos e velozes, além de possuir, dentre as oferecidas, não só uma biblioteca de células já convalidadas, mas, também, o “design kit” pronto, possibilitando atingir o “layout” final de forma automática. A obtenção deste “design kit” pela UNESP foi realizada através do CMP (Circuit Multi Projects), da França, com custo gratuito para a Universidade.

Neste estágio do projeto ainda não se partiu para a obtenção do “layout” de qualquer um dos circuitos aqui apresentados e simulados, e nem mesmo para a integração do conjunto deles. O principal motivo para que isto acontecesse foi a conclusão de que a integração, nesta etapa, teria como principal função o teste da tecnologia escolhida, uma vez que para o projeto não traria um retorno significativo no momento. Esta conclusão foi tirada da experiência adquirida quando do desenvolvimento do LOGA [10-11], onde foram integrados, em tecnologia CMOS 2 μm , 2 dispositivos que continham os circuitos de controle dos seus “pipelines”. Deste modo, optou-se, enquanto continua o desenvolvimento do dispositivo microcontrolador, sintetizá-lo utilizando dispositivos lógicos programáveis (DLP), o que pode vir a ser feito dentro do próprio Campus de Guaratinguetá. Esta estratégia permitirá tomar contato com uma outra forma de implantação de um ASIC, e trará uma experiência nova neste campo. Porém, deve-se, neste ponto, alertar para o problema da execução dos testes do dispositivo, uma vez o mesmo pronto. Até onde se tenha conhecimento, não existe equipamento disponível, com acesso pela Universidade, que possibilite os testes necessários para este tipo de dispositivos no Brasil. Talvez se possa tentar realizar uma parte destes testes utilizando os laboratórios do Centro de Tecnologia para Informática (CTI), em Campinas, porém, no momento, não será possível testá-lo como um todo.

Para demonstrar como o pacote de “software” é poderoso, é apresentado na figura VI.1 a síntese do esquemático do controlador das três máquinas. Para a geração deste esquemático utilizou-se o “software” SYNERGE, do pacote da CADENCE. A geração do esquemático foi realizado nos



laboratórios da QUICKCHIP Engenharia e Projetos Eletrônicos Ltda, representante da CADENCE no Brasil, em função do pacote existente no Campus de Guaratinguetá, da UNESP, não possuir licença para uso deste programa.

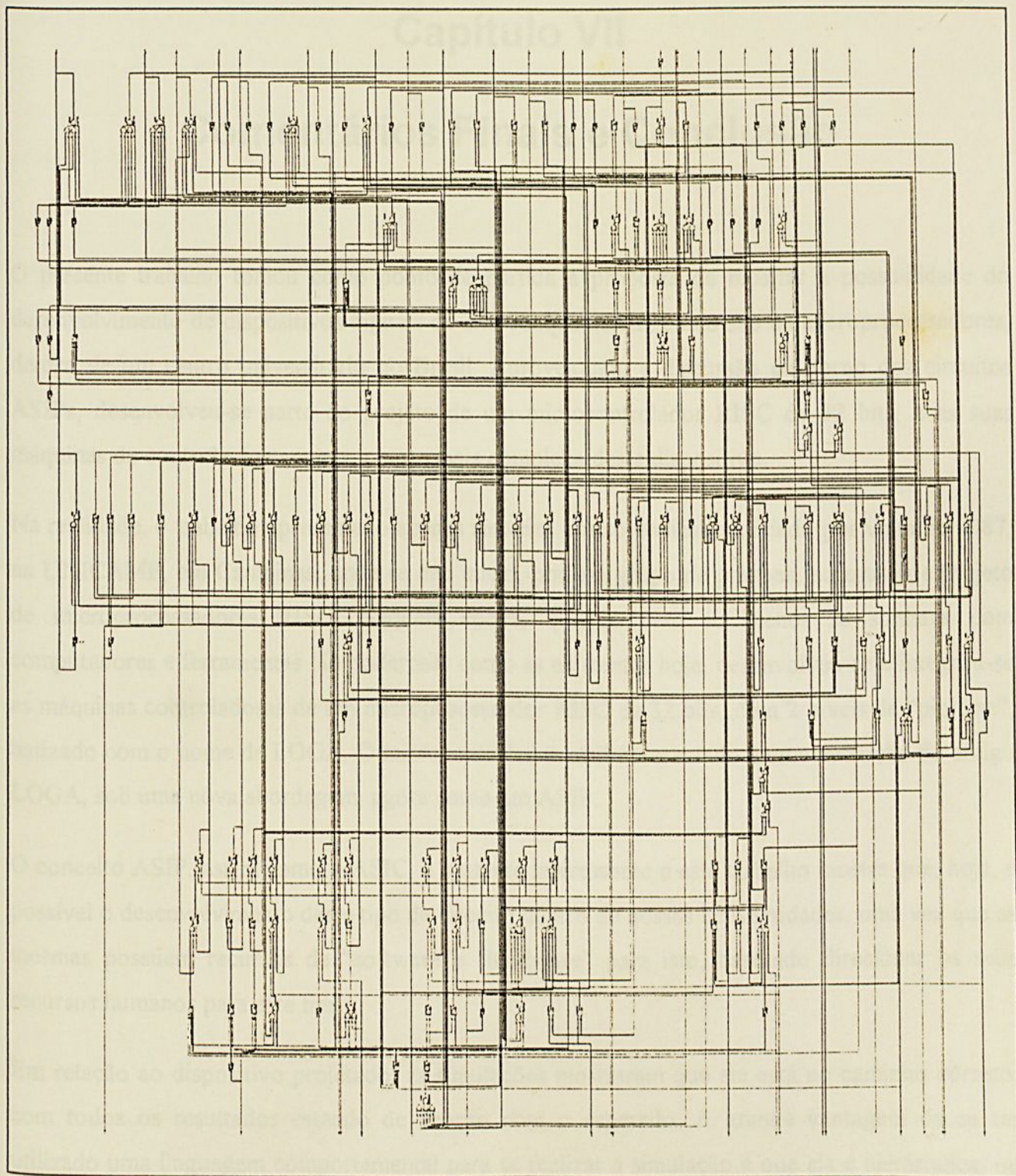


Figura VI.1 - Diagrama esquemático do microcontrolador

Capítulo VII

Comentários Finais e Conclusão

O presente trabalho tomou como ponto de partida a proposta de mostrar a possibilidade do desenvolvimento de dispositivos digitais de alta complexidade, como são os microprocessadores, dentro de um centro universitário no Brasil. Aproveitando a discussão em torno dos circuitos ASIPs, desenvolveu-se parte do projeto de um microcontrolador RISC de 32 bits, e as suas máquinas de controle, justamente a parte mais complexa deste dispositivo.

Na realidade, o trabalho apresentado apenas sintetiza um aprendizado iniciado por volta de 1987, na UNICAMP, em Campinas, onde se deu início, por um grupo de pessoas, ao estudo e projeto de microprocessadores RISC. Naquela época, quando nem ao menos se sonhava com computadores e ferramentas tão poderosas como as existentes hoje, desenvolveu-se e integrou-se as máquinas controladoras de um microprocessador RISC de 32 bits, com 2 níveis de “pipeline”, batizado com o nome de LOGA. O microcontrolador aqui apresentado é uma evolução do antigo LOGA, sob uma nova abordagem, agora como um ASIP.

O conceito ASIP, assim como o ASIC, é bastante interessante e este trabalho mostra que, hoje, é possível o desenvolvimento deste tipo de circuito dentro de nossas Universidades, uma vez que as mesmas possuem recursos de “software” e “hardware” para isto, bastando direcionar os seus recursos humanos para este fim.

Em relação ao dispositivo projetado, as simulações mostraram que ele está no caminho correto, com todos os resultados estando de acordo com o esperado. A grande vantagem de se ter utilizado uma linguagem comportamental para se realizar a simulação é que ela é hierárquica, ou seja, da mesma forma que foi feito com o “flip-flop”, transformando-o em um bloco para utilizá-lo



posteriormente, pode-se fazer com as máquinas de controle dos “pipelines” ou com cada bloco que se vai adicionando ao projeto, até o mesmo estar completo e simulado no seu todo.

Uma vez isto feito pode-se sintetizar de forma automática o seu esquemático, para análises posteriores que se desejar fazer. Tendo sido concluídas todas as análises, a geração do “layout” de integração é, também, automática. É importante ressaltar neste ponto que, para o circuito desenvolvido, não foram implementados circuitos para a sua testabilidade, o que deverá ser feito posteriormente.

Não se pretende aqui sugerir temas e assuntos para continuidade do mesmo, pois só implementação dos outros blocos que o constitui, bem como a sua obtenção na forma final, já constitui um grande desafio a todos aqueles que tiverem interesse em contribuir para o seu desenvolvimento.



Símbolos Utilizados

di - Entrada de dados

clk - Entrada do pulso de clock

clr - Entrada clear

pr - Entrada preset

q - Saída

PC - "TIPO"

J - Instruções de salto ou chamadas

R - Instrução de retorno

Descrição dos Circuitos em Linguagem Verilog - HDL e dos Estímulos Aplicados

bc - Avaliação da condição para saltos ou chamadas

Call - Instrução de chamada a subrotina

Mask - Permissão de acesso aos registradores

Devios - Instrução de desvio, chamada ou retorno

Save - Instrução de salvamento de dados

Via RFB - Instrução de salvamento empregada pelo registro RFB

Load - Instrução de carregamento de dados

I - Carregamento de dados imediatos

Op. Regist. - Instrução de operações entre registros

Shift - Operação de deslocamento de dados

DS - Fila para armazenamento da máquina C



Simbologia Utilizada

dt	- Entrada de dados
clk	- Entrada do pulso de clock
clr	- Entrada clear
pr	- Entrada preset
q	- Saída
FC _A	- "FIFO" da máquina A cheia
J	- Instruções de salto ou chamadas
R	- Instrução de retorno
EF	- Instruções de salto ou chamada com endereçamento efetivo
Cond	- Saltos ou chamadas condicionais
b0	- Avaliação da condição para saltos ou chamadas
Call	- Instrução de chamada a subrotinas
Mackb	- Permissão de acesso aos barramentos
Desvios	- Instruções de salto, chamada ou retorno
Save	- Instrução de salvamento de dados
Via RFB	- Instrução de salvamento endereçadas pelo registro RFB
Load	- Instrução de carregamento de dados
I	- Carregamento de dados imediatos
Op. Regist.	- Instrução de operações entre registros
Desloc.	- Operação de deslocamento de dados
DS.	- Fila para armazenamento da máquina C



Circuito do Gerenciador de Acesso aos Barramentos

a) Descrição do Circuito

```
module geren (Mrqa,Mrqb,Mrqc,Mbusya,Mbusyb,Macka,Mackb,Mackc);
input Mrqa,Mrqb,Mrqc,Mbusya,Mbusyb;
output Macka,Mackb,Mackc;
and    ad1 (Macka,Mrqa,nMrqb,nMrqc),
        ad2 (Mackb,Mrqb,nMrqc,nMbusya),
        ad3 (Mackc,Mrqc,nMbusya,nMbusyb);
not     iv1 (nMrqb,Mrqb),
        iv2 (nMrqc,Mrqc),
        iv3 (nMbusya,Mbusya),
        iv4 (nMbusyb,Mbusyb);
endmodule
```

b) Descrição dos estímulos aplicados as entradas

```
initial
begin
    Mrqa = 1;
    Mrqb = 0;
    Mrqc = 0;
    Mbusya = 0;
    Mbusyb = 0;
#100 Mrqa = 0;
    Mrqb = 1;
#100 Mrqb = 0;
    Mrqc = 1;
#100 Mrqa = 1;
    Mrqb = 1;
    Mrqc = 0;
```



```
#100 Mrqc = 1;  
#100 Mrqc = 0;  
      Mbusya = 1;  
#100 Mrqc = 1;  
end
```

a) Descrição do circuito do Flip-Flop

```
module flop (dt, clk, clr, pr, q),  
  input  dt, clk, clr, pr,  
  output q;  
  nand  nd1 (a, dt, clk, clr),  
        nd2 (b, nd1, clk, pr),  
        nd3 (c, a, d, pr),  
        nd4 (d, a, b, clr),  
        nd5 (e, c, nd4),  
        nd6 (f, d, nd5),  
        nd7 (g, e, g, pr),  
        nd8 (g, q, f, clr),  
  not   n1 (nd1, dt),  
        n2 (nd6, nd5),  
endmodule
```

b) Descrição dos estímulos aplicados as entradas

```
always  
initial  
begin  
#5   clk = 0;  
#5   clk = 1;  
end  
initial  
begin  
    dt = 0;  
    clr = 1;  
    pr = 1;  
end
```


“Flip-Flop” utilizado na descrição das máquinas de estado

a) Descrição do circuito do Flip-Flop

```
module flop (dt, clk, clr, pr, q);  
input  dt, clk, clr, pr;  
output q;  
nand  nd1 (a, dt, clk, clr),  
      nd2 (b, ndt, clk, pr),  
      nd3 (c, a, d, pr),  
      nd4 (d, c, b, clr),  
      nd5 (e, c, nclk),  
      nd6 (f, d, nclk),  
      nd7 (q, e, g, pr),  
      nd8 (g, q, f, clr);  
not    n1 (ndt, dt),  
       n2 (nclk, clk);  
endmodule
```

b) Descrição dos estímulos aplicados as entradas

```
always  
initial  
begin  
#5    clk = 0;  
#5    clk = 1;  
end  
initial  
begin  
    dt = 0;  
    clr = 1;  
    pr = 1;
```


#10 dt = 1;

#10 dt = 0;

#10 pr = 0;

#10 dt = 1;

#10 pr = 1;

clr = 0;

#10 dt = 0;

#10 dt = 1;

clr = 1;

end

initial #150 \$finish;

Máquina de Busca

a) Descrição do circuito de controle

```
module maqbusca (da, clk, pr, clr, qa, qb, qc, qd, qe, qf, qg, qh, qi, qj, qk, macka,
                 fca, j, bo, r, ef, call, cond);
input  da, clk, pr, clr, macka, fca, j, bo, r, ef, call, cond;
output qa, qb, qc, qd, qe, qf, qg, qh, qi, qj, qk;
flop   fa (da, clk, pr, clr, qa),
       fb (db, clk, clr, pr, qb),
       fc (dc, clk, clr, pr, qc),
       fd (dd, clk, clr, pr, qd),
       fe (de, clk, clr, pr, qe),
       ff (df, clk, clr, pr, qf),
       fg (dg, clk, clr, pr, qg),
       fh (dh, clk, clr, pr, qh),
       fi (di, clk, clr, pr, qi),
       fj (dj, clk, clr, pr, qj),
       fk (dk, clk, clr, pr, qk);
or     o1 (db, qa, a, b),
       o2 (e, c, d, qd, qe, qk),
       o3 (f, qf, qg),
       o4 (di, g, h),
       o5 (dk, l, qj);
and    a1 (a, nmacka, qb),
       a2 (b, nfca, e),
       a3 (c, nj, qc),
       a4 (d, nbo, qh),
       a5 (dc, macka, qb),
       a6 (dd, fca, e),
       a7 (de, j, r, qc),
       a8 (df, j, nr, ef, qc),
       a9 (dg, j, nr, nef, qc),
       a10 (dh, cond, f),
```




```
#10    a11 (g, ncond, f),
#10    a12 (h, bo, qh),
#20    a13 (dj, call, qi),
        a14 (l, ncall, qi);
not    n1 (nmacka, macka),
#20    n2 (nfca, fca),
#10    n3 (nj, j),
#10    n4 (nbo, bo),
        n5 (nr, r),
#10    n6 (nef, ef),
#10    n7 (ncond, cond),
#10    n8 (ncall, call);
endmodule
```

b) Descrição dos estímulos aplicados as entradas

```
always
initial
begin
#5    clk = 0;
#5    clk = 1;
end
initial
begin
    da = 0;
    clr = 0;
    pr = 1;
    mackc = 0;
    fca = 0;
    j = 0;
    bo = 0;
    r = 0;
    ef = 0;
    cond = 0;
    call = 0;
#10   clr = 1;
#20   mackc = 1;
```




```
#10 fca = 1;
#10 fca = 0;
#20 j = 1;
    r = 1;
#10 r = 0;
#20 ef = 1;
#10 cond = 1;
#10 bo = 1;
    cond = 0;
#10 call = 1;
#10 call = 0;
#40 ef = 0;
#10 cond = 1;
#50 bo = 0;
    cond = 0;
#20 j = 0;
#10 macka = 0;
end
initial #300 $finish;
```



Máquina de Execução:

a) Descrição do circuito

```
module maqexec (da, clk, pr, clr, qa, qb, qc, qd, qe, qf, qg, qh, qi, qj, qk, ql,
                mackb, desvios, noinst, load, viarfb, save, ds, i, opreg, desloc);
input  da, clk, pr, clr, mackb, desvios, noinst, load, viarfb, save, ds, i, opreg, desloc;
output qa, qb, qc, qd, qe, qf, qg, qh, qi, qj, qk, ql;
flop   fa (da, clk, pr, clr, qa),
       fb (db, clk, clr, pr, qb),
       fc (dc, clk, clr, pr, qc),
       fd (dd, clk, clr, pr, qd),
       fe (de, clk, clr, pr, qe),
       ff (df, clk, clr, pr, qf),
       fg (dg, clk, clr, pr, qg),
       fh (dh, clk, clr, pr, qh),
       fi (di, clk, clr, pr, qi),
       fj (dj, clk, clr, pr, qj),
       fk (dk, clk, clr, pr, qk),
       fl (dl, clk, clr, pr, ql);
and    a1 (a, desvios, qb),
       a2 (b, noinst, qb),
       a3 (dc, load, i, qb),
       a4 (c, load, ni, qb),
       a5 (d, nmackb, qd),
       a6 (de, mackb, viarfb, qd),
       a7 (df, mackb, nviarfb, qd),
       a8 (e, save, viarfb, qb),
       a9 (f, ds, qh),
       a10 (g, nds, qh),
       a11 (h, nds, qj),
       a12 (j, save, nviarfb, qb),
       a13 (k, ds, qj),
       a14 (dk, opreg, desloc, qb),
```




```
a15 (dl, opreg, ndesloc, qb);
or    o1 (db, qa, a, qc, qg, qi, qk, ql, b),
      o2 (dd, c, d),
      o3 (dg, qe, qf),
      o4 (dh, e, f),
      o5 (di, g, h),
      o6 (dj, j, k);
not   n1 (ni, i),
      n2 (nviarfb, viarfb),
      n3 (nds, ds),
      n4 (ndesloc, desloc);
      n5 (nmackb, mackb);
endmodule
```

b) Descrição dos estímulos aplicados as entradas

```
always
initial
begin
#5    clk = 0;
#5    clk = 1;
end
initial
begin
    da = 0;
    clr = 0;
    pr = 1;
    noinst = 1;
    desvios = 0;
    opreg = 0;
    desloc = 0;
    save = 0;
    mackb = 0;
    viarfb = 0;
    ds = 0;
    i = 0;
    load = 0;
```




```
#10  clr = 1;
#20  desvios = 1;
      noinst = 0;
#10  desvios = 0;
      i = 1;
      load = 1;
#10  i = 0;
#30  mackb = 1;
#40  viarfb = 1;
      load = 0;
#30  save = 1;
#10  ds = 1;
#10  ds = 0;
#20  viarfb = 0;
#10  save = 0;
#20  opreg = 1;
#20  desloc = 1;
#20  opreg = 0;
      desloc = 0;
      noinst = 1;
end
initial #300 $finish;
```


Máquina de Armazenamento

a) Descrição do circuito

```
module maqarm (da, clk, pr, clr, qa, qb, qc, qd, qe, fvc, mackc);
input  da, clk, pr, clr, fvc, mackc;
output qa, qb, qc, qd, qe;
flop   fa (da, clk, pr, clr, qa),
       fb (db, clk, clr, pr, qb),
       fc (dc, clk, clr, pr, qc),
       fd (dd, clk, clr, pr, qd),
       fe (de, clk, clr, pr, qe),
and    a1 (a, nfvc, qb),
       a2 (b, nmackc, qc),
       a3 (dd, mackc, qc),
       a4 (de, qd),
       a5 (c, fvc, qb);
or     o1 (db, qa, c, qe),
       o2 (dc, a, b),
not    n1 (nfvc, fvc),
       n2 (nmackc, mackc);
endmodule
```

b) Descrição dos estímulos aplicados ao circuito

```
always
initial
begin
#5    clk = 0;
#5    clk = 1;
end
initial
begin
    da = 0;
    clr = 0;
```




```
pr = 1;  
fvc = 1;  
mackc = 0;  
#10 clr = 1;  
#20 fvc = 0;  
#20 mackc = 1;  
#40 mackc = 0;  
end  
initial #150 $finish;
```


Circuito de Controle do Microcontrolador

a) Descrição do circuito

```
module controle (da, clk, pr, clr, qaa, qba, qca, qda, qea, qfa, qga, qha, qia, qja, qla,
    qka, fca, j, bo, r, ef, call, cond, qab, qbb, qcb, qdb, qeb, qfb, qgb, qhb,
    qib, qlb, desvios, noinst, load, viarfb, save, ds, i, opreg, desloc, qac,
    qbc, qcc, qdc, qec, fvc);
input  da, clk, pr, clr, fca, j, bo, r, ef, call, cond, desvios, noinst, load, viarfb, save,
    ds, i, opreg, desloc, fvc;
output qaa, qba, qca, qda, qea, qfa, qga, qha, qia, qja, qla, qka, qab, qbb, qcb,
    qdb, qeb, qfb, qgb, qhb, qib, qlb, qac, qbc, qcc, qdc, qec;
maqbusca    ma (da, clk, pr, clr, qaa, qba, qca, qda, qea, qfa, qga, qha, qia, qja, qla,
    qka, macka, fca, j, bo, r, ef, call, cond);
maqexec      mb (da, clk, pr, clr, qab, qbb, qcb, qdb, qeb, qfb, qgb, qhb, qib, qlb,
    mackb, desvios, noinst, load, viarfb, save, ds, i, opreg, desloc);
maqarm       mc (da, clk, pr, clr, qac, qbc, qcc, qdc, qec, mackc, fvc);
geren  ge (qba, qdb, qcc, qca, mbusyb, macka, mackb, mackc);
and    a1 (mbusyb, qeb, qfb, qgb);
endmodule
```

l) Descrição dos estímulos aplicados as entradas

```
always
initial
begin
#5    clk = 0;
#5    clk = 1;
end
initial
begin
    da = 0;
    clr = 0;
    pr = 1;
    fca = 0;
```




```
j = 0;
bo = 0;
r = 0;
ef = 0;
cond = 0;
call = 0;
noinst = 1;
desvios = 0;
opreg = 0;
desloc = 0;
save = 0;
viarfb = 0;
ds = 0;
i = 0;
load = 0;
fvc = 1;
#10 clr = 1;
#20 noinst = 0;
load = 1;
#20 load = 0;
noinst = 1;
#30 load = 1;
i = 1;
noinst = 0;
#10 load = 0;
i = 0;
#10 opreg = 1;
#10 opreg = 0;
#10 save = 1;
viarfb = 1;
#10 save = 0;
viarfb = 0;
#10 fvc = 0;
#10 opreg = 1;
desloc = 1;
fvc = 1;
#10 j = 1;
```



- desvios = 1;
#10 j = 0;
desvios = 0;
noinst = 1;
#30 clr = 0;

Referências Bibliográficas

- [1] - M., "Survey of Advanced Microprocessors", Van Nostrand Reinhold, New York, 1991.
- [2] - Lee, W. Et al, "A 1V DSP for Wireless Communications", IEEE International Solid State Circuits Conference, February, 6-8, 1997, S. Francisco, California, USA.
- [3] - Martins, T. Et al, "A 5-Level Storage 40 DRAM", IEEE International Solid State Circuits Conference, February, 6-8, 1997, S. Francisco, California, USA.
- [4] - Shimomura, K. Et al, "A 40V 50T DRAM with Body Control Technique", IEEE International Solid State Circuits Conference, February, 6-8, 1997, S. Francisco, California, USA.
- [5] - Orsaka, R, "A 500MHz Superscalar RISC Microprocessor with Out-of-Order Execution", IEEE International Solid State Circuits Conference, February, 6-8, 1997, S. Francisco, California, USA.
- [6] - Cohen, E., "A 330MHz Superscalar Microprocessor", IEEE International Solid State Circuits Conference, February, 6-8, 1997, S. Francisco, California, USA.
- [7] - Hsieh, K. & Tzeng, D., "Microprocessors: Architecture, Implementation, & Programming", McGraw-Hill, 1992.
- [8] - Markolf, J., "RISC (MIPS)", 1-16, novembro de 1991.
- [9] - Patterson, D. A., "Reduced Instruction Set Computer", Comm. of the ACM, janeiro de 1974.
- [10] - Botura, Jr., Pereira, L.O., Chaves, N.P., Pires, F.R.S. "LDA301: Microprocessador RISC de 32 Bits com Otimização de Margem de Ruído", Anais do IV Congresso de Sociedade Brasileira de Microeletrônica, pg. 266-274, 12 a 14 de julho de 1999, Porto Alegre, RS.
- [11] - Botura, Jr., Pereira, L.O., Chaves, N.P., Pires, F.R.S. "LDA301: Proposta de um Microprocessador RISC de 32 Bits com Otimização RISC", Congresso de Sociedade Brasileira de Microeletrônica, pg. 275-284, 12 a 14 de julho de 1999, S. Paulo, SP.



Referências Bibliográficas

- [1] - Veronis, A. M.; "Survey of Advanced Microprocessors", Van Nostrand Reinhold, New York, 1991
- [2] - Lee, W. Et al; "A 1V DSP for Wireless Communication", IEEE International Solid State Circuits Conference, February, 6-8,1997, S.Francisco, California, USA
- [3] - Murotani, T Et al; "A 1-Level Storage 4Gb DRAM", IEEE International Solid State Circuits Conference, February, 6-8,1997, S.Francisco, California, USA
- [4] - Shimomura, K Et al; "A 46ns SOI-DRAM with Body Control Technique", IEEE International Solid State Circuits Conference, February, 6-8,1997, S.Francisco, California, USA
- [5] - Gieseke, B; "A 600MHz Superscalar RISC Microprocessor with Out-of-Order Execution", IEEE International Solid State Circuits Conference, February, 6-8,1997, S.Francisco, California, USA
- [6] - Cohen, E.; "A 533MHz Superscalar Microprocessor ", IEEE International Solid State Circuits Conference, February, 6-8,1997, S.Francisco, California, USA
- [7] - Hintz, K. & Tabak, D.; Microcontrollers: "Architecture, Implementation, & Programming", McGraw-Hill, 1992.
- [8] - Markoff, J.; "RISC CHIPS"; Byte; novembro de 1984;
- [9] - Patterson, D.A.; "Reduced Instruction Set Computer"; Communication of the ACM; janeiro de 1985.
- [10] - Botura, Jr.; Ferreira, L.O.; Canetti, N.F; Pedro, P.H.S "LOGA01: Microprocessador RISC de 32 Bits - Detalhamento da Máquina de Busca"; Anais do IV Congresso da Sociedade Brasileira de Microeletrônica, pg 286-294, 12 a 14 de julho de 1989, Porto Alegre, RS.
- [11] - Botura, Jr.; Ferreira, L.O.; Canetti, N.F; Pedro, P.H.S "LOGA01/LED :- Proposta de um Microprocessador de 32 bits utilizando arquitetura RISC" III Congresso da Sociedade Brasileira de Microeletrônica, pg 355-364, 12 a 14 de julho de 1988, S.Paulo, SP..



- [12] - Botura, Jr.; Ferreira, L.O.; Canettieri, N.F.; Pedro, P.H.S.; Ferreira, L.O.;
"Integração da Máquina de busca de um microprocessador de 32 bits no
LED/UNICAMP"; Anais do X Seminário da ADUNESP, pg 304 a 313, 20 a 23 de
novembro de 1989, Guaratinguetá, SP.
- [13] - Yaglica, J.J. and Gilmour, P.S. "How to Select a microcontroler", IEEE Spectrum,
pp 106-109, novembro de 1990.
- [14] - Patterson, D.A., and Séquin, C.H. "A VLSI RISC", Computers, Vol 15, no 9, pp8-
21, setembro de 1982
- [15] - Katevenis, M.G.H; "The RISC II Microarchitecture", Proc. VL:SI'83, inter. Conf.
Varey Large Scale Integration, Trondheim, Norway, Agosto de 1983, PP 349-359.
- [16] - Ito, R.; "Estudo, Proposta e Projeto de um Microcontrolador Integrado"; Relatório
de iniciação científica enviado a FAPESP (processo 92/1121-8), 1992-1993.
- [17] - Salvan Jr, C.E.; "Estudo, Simulação e Projeto de um Microcontrolador Integrado
com Arquitetura RISC"; Relatório de iniciação científica enviado a FAPESP
(processo 95/6664-8), 1996.
- [18] - CTI; Quickchip ; "Uma visão da fabricação, projeto e aplicação de circuitos
integrados de aplicação específica"; Julho 1993.
- [19] - Sociedade Brasileira de Microeletrônica; "Formas de Implementação de ASICs
Digitais"; Setembro 1993.
- [20] - Verilog-XL Reference Manual; Volume I, II; Version 1.6. March 1991.
- [21] - Design Entry Reference Manual; Volume I, Cap. 5, Version 4.2. October 1991.



