UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA DE CAMPINAS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ARQUITETURA DE UM MICROCOMPUTADOR PARA CONTROLE DE PROCESSOS CONCORRENTES EM TEMPO REAL

TESE DE MESTRADO

AUTOR : Paulo Lício de Geus

ORIENTADOR : Prof.Dr. Mário Jino

Este exemplar corresponde à redação final da tese defendida por Paulo Lício de Geus e aprovada pela Comissão Julgadora em 19/07/1985.

Mario Jino

Julho de 1985

RESUMO

Este trabalho descreve modificações introduzidas sobre a arquitetura existente de um microcomputador modular para atender requisitos de novas aplicações propostas. Tais aplicações envolvem aquisição de dados e processamento em tempo real, além do gerenciamento de um grande número de interfaces de comunicação série. O objetivo procurado foi a obtenção de um sistema com baixo tempo de resposta a eventos assíncronos, otimização do fluxo de dados e boa taxa de processamento, dentro das limitações de gerenciamento de memória implícitas à CPU de 8 bits utilizada.

As principais modificações de hardware introduzidas são: uma unidade de gerenciamento de memória com páginas de 4 Kbytes e 64 contextos de paginação residentes; um esquema de interrupções vetorizadas (família Z80) para a identificação automática de até 128 eventos diferentes; um controlador de DMA dispondo de 11 canais independentes com acesso ao espaço total de 1 Mbyte do microcomputador.

ABSTRACT

This work describes enhancements made over the existing architecture of a modular microcomputer to support the requirements of proposed new applications. Such applications involve data aquisition and real-time processing, as well as the management of several serial communication interfaces. The desired goal was to get a system with low response time to asynchronous events, optimized data flow and good throughput, within the memory management limitations implied by the use of an 8 bit CPU.

The main hardware alterations are: a memory-management unit with 4K byte pages and 64 resident pagination contexts; a vectorized interrupt scheme (Z80 family) for automatic identification of up to 128 different events; a DMA controller supporting 11 independent channels, with access to the full 1 Mbyte space of the microcomputer.

AGRADECIMENTOS

Ao Mário Jino, pela orientação.

Ao Cléslo Tozzi, pela análise crítica.

Ao Célio Guimarães, pela idéia.

Ao Rubens Machado, "pai" do sistema.

Aos colegas da P&D, que de alguma forma contribuíram para a realização deste trabalho: Arnaldo, Elio, Glicério, Harada, Jorge Omar, Juceley, Leonardo, Renato, Rubia, Sandra, Sidnei, Wolfgang.

A P&D Sistemas Eletrônicos S/A, ambiente deste trabalho.

à Cristiane

ÍNDICE

ÍNDICE DE FIGURAS

pág.

CAPÍTULO 0

INTRODUÇÃO

Microprocessadores têm sido largamente utilizados em controle de processos, geralmente na aquisição de dados e atuação sobre o processo, graças à simplicidade do hardware necessário. A um nível superior, porém, vários processos precisam ser executados concorrentemente, muitas vezes sujeitos a requisitos de tempo real, requerendo uma arquitetura de microcomputador mais elaborada que as usualmente utilizadas.

Este trabalho descreve as modificações introduzidas sobre a arquitetura existente de um microcomputador modular para atender requisitos de novas aplicações propostas. Tais aplicações envolvem um grande volume de dados adquiridos e processamento, como também o gerenciamento de um grande número de interfaces de comunicação [PDS183]. O objetivo procurado foi a obtenção de um sistema com baixo tempo de resposta a eventos assíncronos, otimização do fluxo de dados e boa taxa de processamento, dentro das limitações de gerenciamento de memória para ampliação do espaço de endereçamento, implícitas à CPU de 8 bits utilizada.

As principais modificações de hardware introduzidas são: uma unidade de gerenciamento de memória com páginas de 4 Kbytes e 64 contextos de paginação residentes na unidade, cuja mudança é executada em menos de 3 us; um esquema de interrupções vetorizadas (família Z80) para a identificação

automática de até 128 eventos diferentes; um controlador de DMA dispondo de 11 canais independentes com acesso ao espaço total de 1 Mbyte do microcomputador, um deles capaz de transferências memória à memória.

O microcomputador assim modificado é utilizado por um sistema de supervisão de centrais telefônicas em dois níveis hierárquicos. Graças à sua modularidade [JENS74] basta reconfigurar suas placas para se obter o controlador desejado. Uma breve descrição das tarefas e complexidades dos controladores destes dois níveis é apresentada, juntamente com medidas da taxa de ocupação dos sistemas mais carregados. São descritas também outras aplicações do sistema, já implementadas ou em fase de desenvolvimento.

No capítulo 1 tem-se um histórico do desenvolvimento do sistema, descrevendo-se o sistema básico inicial e o sistema de supervisão (chamado SITASU) que o utilizava em seu núcleo. A implementação do segundo nível hierárquico de supervisão e as funções acrescentadas ao primeiro nível apontaram a necessidade de torná-lo mais poderoso, sendo analisadas as alternativas para se obter isto: arquitetura multi-processada, CPU de 16 bits e reformulação da arquitetura existente. A alternativa escolhida (reformulação) é justificada, citando-se as características adicionadas, a localização das mudanças e os objetivos procurados.

O capítulo 2 descreve o esquema de gerenciamento de memória, partindo do problema básico de ampliação do espaço de

endereçamento da CPU e seguindo com a discussão das alternativas para se obtê-la: segmentação, mapeadores [SPRY81], segmentos variáveis contíguos [BRON82] e paginação [GUIM79]. Justifica-se a adoção do esquema de paginação, citando-se as facilidades do Z80 para implementá-lo. Segue-se a resolução dos problemas de hardware até a definição do circuito, descrevendo-se a operação da MMU (unidade de gerenciamento) e analisando-se a perda de desempenho ocasionada. Suas principais características são: tabela de paginação com 16 entradas e páginas de 4 Kbytes, mapeando um espaço lógico de 64 Kbytes em um espaço físico de 1 Mbyte; manutenção de 64 tabelas estáticas na memória da MMU; tempo para mudança de contexto (tabela) menor que 3 us.

O capítulo 3 descreve o esquema de interrupções vetorizadas implementado. Levando-se em conta as características extras que um esquema de interrupções para aplicações em tempo real deve ter [PRAS70], parte-se do esquema existente e compara-se com o esquema da família Z80, apontando-se os benefícios que a utilização do "chip" Z80-SIO pode trazer sob este esquema, devido ao grande número de canais série que poderia estar presente nas aplicações previstas. Detalha-se também o funcionamento da cadeia de prioridades adicionada ao barramento que apóia o esquema, além da emulação provida pelo "chip" Z80-CTC aos periféricos não pertencentes à família Z80 [ZILO81].

No capítulo 4 são comparados os tempos de resposta e de ocupação da CPU, gerenciando um grande número de canais

série, utilizando "polling", interrupções vetorizadas e DMA. Segue-se uma descrição do hardware da placa controladora de DMA suportando 11 canais.

No capítulo 5 são brevemente descritos os controladores (UCR, USCE) utilizados no sistema de supervisão de centrais telefônicas (SITASU), procurando destacar a complexidade dos mesmos. Medidas de desempenho foram realizadas nas unidades instaladas sujeitas a cargas maiores, comprovando o atendimento completo das funções requeridas pelos sistemas (em particular os requisitos de tempo real) sem esgotamento da capacidade de processamento da CPU.

No capítulo 6 são apresentadas novas aplicações para este microcomputador, destacando sua versatilidade: um sistema de processamento multiusuário, já em funcionamento; um sistema servidor de arquivos, em fase de desenvolvimento; e um sistema de telessupervisão tolerante a falhas (em fase de desenvolvimento), onde a maior preocupação é a redundância nos nodos da rede de interconexão e a degradação lenta em caso de falhas.

No capítulo 7 são apresentadas conclusões sobre o trabalho realizado.

CAPÍTULO 1

HISTÓRICO E COLOCAÇÃO DO PROBLEMA

Este capítulo apresenta um histórico do desenvolvimento do sistema, analisa alternativas para se aumentar o desempenho do sistema e discute a alternativa adotada, apresentando uma visão geral das modificações.

1.1 Ambiente de utilização do sistema

O desenvolvimento do sistema estudado aqui, realizado junto à P&D Sistemas Eletrônicos S/A de Campinas SP, foi motivado pela necessidade de controladores configurados a diversas aplicações diferentes que, contudo, não podiam ser dedicados a apenas uma função, em vista do pequeno número de equipamentos produzidos de cada tipo. Estes equipamentos são geralmente sistemas de supervisão e controle de processos, nos quais a maior parte dos custos incide sobre as interfaces de monitoração e atuação sobre o processo. Modularidade aqui é um dos aspectos mais críticos [JENS74]: antes de mais nada, cada aplicação é geralmente única; além disso, à época do desenvolvimento/instalação do equipamento muitas vezes os requisitos de computação estão mal definidos, estando sujeitos a redefinições ao longo da vida do sistema. O computador deve portanto acomodar incrementos funcionais sem causar grande impacto sobre o hardware ou o software. Deste modo, optou-se por um sistema baseado em microprocessadores cujos recursos eram

adicionados modularmente, na forma de placas funcionais que se comunicam através de um barramento padronizado.

1.2 Descrição do barramento (via SME)

A definição deste barramento previu a utilização inicial de CPU's de 8 bits, admitindo substituição futura por CPU's de 16 bits assim que novas aplicações exigissem maior capacidade de processamento. Já que os sinais particulares do microprocessador de uma placa de CPU devem necessariamente ser conformados aos sinais padronizados do barramento, todas as outras placas (de memória e dos diversos dispositivos de E/S) devem manter operação normal independente de a CPU utilizada ser de 8 ou 16 bits.

Este barramento padronizado, denominado via SME (sistema SME), utiliza conectores de 100 pinos, dos quais 22 não pertencem ao barramento, mas são utilizados na conexão de sinais de placas do SME com dispositivos externos ao sistema e de sinais extra-barramento entre placas do SME. As 100 vias do conector podem ser agrupadas nas seguintes categorias de sinais:

- 20 linhas de endereço: BE0-BE19

- 16 linhas de dados: BD0-BD15

- 8 linhas de interrupção: BINT0-BINT7

- 14 linhas de controle:

  - leitura de E/S: /LES
  - escrita em E/S: /EES
  - leitura de memória: /LEM
  - escrita em memória: /ESM
  - hab. byte mais significativo: /HBM
  - inicialização: /INIC
  - relógio: REL
  - referência: /REF
  - fim de transferência: /FTR
  - reconhecimento interrupção: /RIN
  - entrada de prioridade: /EPR
  - saída de prioridade: /SPR
  - requisição da via: /HOLD
  - cessão da via: /HLDA

- 20 linhas de alimentação: +5V, +12V, -12V, GND

- 22 linhas de uso geral, não pertencentes ao barramento.

O tamanho (padronizado) de placas do sistema é de 156 x 160 mm. As transferências de dados são realizadas seguindo um protocolo assíncrono, deslocando 8 ou 16 bits de cada vez, a uma taxa de 2 Mtransf/s.

1.3 Sistema SITASU

1.3.1 Hardware básico

O sistema inicialmente implementado era constituído por uma CPU 8085 ou Z80, 64 Kbytes de memória RAM, 16 Kbytes de memória EPROM, 2 canais de interface série, controlador de diskette, controlador de DMA e interfaces para 2 tipos de sistemas de aquisição de dados.

Uma destas interfaces é genérica, fazendo a conexão com um conjunto de gavetas (Sistema TAJUS) cujas placas são exclusivamente dedicadas a funções de E/S (aquisição de dados e atuação). Como exemplos podemos citar funções de entradas e saídas digitais, entradas e saídas analógicas, além de outras mais específicas relacionadas aos processos encontrados em centrais telefônicas. Esta interface está voltada a eventos relativamente estáticos, normalmente sem grandes requisitos de tempo real.

1.3.2 Hardware dedicado

A segunda destas interfaces é dedicada a um sistema de supervisão de centrais telefônicas (chamado SITASU), adquirindo dados a uma taxa de 100 Kbytes/s, monitorando até 10.000 pontos de supervisão dos órgãos da central telefônica.

Esta interface, ao contrário da primeira, é deveras exigente quanto a requisitos de tempo real, como se pode ver pela

taxa de aquisição mencionada. Uma lógica na própria interface requisita as transferências de dados à placa de DMA.

1.3.3 Estrutura de supervisão

O sistema SITASU é constituído de três níveis hierárquicos de supervisão de uma rede telefônica. O controlador cuja interface de aquisição é descrita no parágrafo anterior é chamado USCE, e é quem efetivamente adquire dados diretamente da central telefônica, e é aí localizado. O segundo nível é controlado por um sistema chamado UCR, e sua função é concentrar e consolidar dados de centrais telefônicas de uma região, fornecidos por sistemas USCE. No terceiro nível fica o sistema UCS, conectado aos vários sistemas UCR e permitindo a gerência de toda a rede telefônica, enfatizando os dados de tráfego e manutenção da rede.

O sistema do nível inferior (USCE), desenvolvido pela TELERJ, foi industrializado pela P&D. Em uma primeira fase apenas o subsistema de aquisição de dados era de fabricação própria, seguindo-se então com a substituição do microcomputador (antes adquirido de outro fabricante). O microcomputador SME era então utilizado em sua configuração inicial, com CPU 8085 e 64/80 Kbytes de memória. Sua principal característica era a modularidade, garantida pela utilização de um barramento padronizado. Este estudo se refere às modificações introduzidas sobre sua arquitetura básica existente, a fim de lhe dar maior capacidade de

processamento.

1.3.4 Novos requisitos, novas implementações

A razão deste aprimoramento foi a execução de um novo projeto, com novos requisitos: em primeiro lugar, novas funções foram acrescentadas ao sistema do nível inferior (USCE), além da ampliação do número máximo de pontos supervisionados para 16.000, exigindo a introdução de um executivo multitarefas capaz de satisfazer os requisitos de tempo real inerentes ao sistema; em segundo lugar, a implementação do segundo nível hierárquico (sistema UCR), na realidade um concentrador de comunicações com um grande número de interfaces seriais, além de dispositivos de memória de massa de média capacidade (disco flexível e disco rígido), também com certos requisitos de tempo real.

É importante ressaltar aqui que o hardware básico é o mesmo para estes dois tipos de controladores, bastando apenas reconfigurar a equipagem de placas especificamente para cada controlador. Mais ainda: pretendeu-se obter um hardware que cobrisse não apenas estas duas aplicações, mas também outras apenas visualizadas na época, e atualmente já implementadas ou em implementação.

1.4 Alternativas para uma maior capacidade de processamento

Dentre os possíveis caminhos para se obter uma arquitetura aperfeiçoada, capaz de satisfazer aos requisitos das novas

aplicações (genericamente: aumentando o desempenho do sistema global), pode-se citar três alternativas, as quais serão analisadas a seguir. Um fator que influiu decisivamente a favor da alternativa escolhida foi a disponibilidade de software, contabilizada de duas formas: a difícil obtenção de software básico, ao início do projeto, para CPU's de 16 bits, evolução natural na busca de maior capacidade de processamento; a existência do software aplicativo do sistema chamado USCE, já desenvolvido, cuja maior parte poderia ser reaproveitada, desde que mantida a CPU original de 8 bits (8085 - Z80).

1.4.1 Arquitetura multiprocessada

A utilização de processamento distribuído tem se difundido largamente não só em controle de processos mas também em várias outras aplicações, devido principalmente à popularização dos microprocessadores, com a consequente diminuição nos preços e posterior crescimento na potência de processamento. CPU's dedicadas a tarefas específicas são interligadas através de uma rede local (acoplamento remoto ou fraco) ou através de um barramento paralelo de alta velocidade (acoplamento próximo ou forte).

Levando-se em conta o hardware já existente (interfaces específicas, demais funções extras acrescentáveis graças à modularidade do sistema inicial), a solução mais imediata deveria passar pelo barramento padronizado. Porém dois obstáculos se apresentaram derrubando esta alternativa. Um requisito básico para esta arquitetura seria a utilização de

CPU's autônomas, com memória e dispositivos de E/S incorporados. O tamanho de placas adotado, que fornece a grande flexibilidade e modularidade necessárias, torna difícil a implementação destas CPU's autônomas. Além disso, este barramento padronizado teria que ser modificado para permitir um bom gerenciamento de vários processadores (controladores de barramento), significando a adição de novos sinais e conseqüente modificação na pinagem padronizada. Em outras palavras, teria de ser projetado todo um sistema novo, sem a possibilidade de aproveitamento das funções de hardware já implementadas.

1.4.2 CPU de 16 bits

Examinada à luz do trabalho de hardware necessário, esta alternativa se mostrou mais atraente que a anterior, pois os recursos de hardware existentes (placas funcionais) seriam integralmente aproveitados. Bastaria projetar apenas a nova placa funcional de CPU (compatível com o barramento padronizado).

Se do lado do hardware existia esta simplicidade aparente, o mesmo não se podia dizer quanto ao do software. A mudança para uma CPU totalmente incompatível em software com a anterior significaria partir do zero não apenas quanto ao software básico (sistema operacional de disco, executivo multitarefas), mas também quanto ao software aplicativo de tratamento do processo em questão. Esta última afirmação pode parecer estranha no âmbito de sistemas de controle de processos, em sua maioria controlados por minicomputadores,

os quais admitem a utilização de programas gerados em linguagens de alto nível para aplicações deste tipo, apesar dos requisitos de tempo real e de velocidade exigidos. O que acontece é que a potência de processamento disponível compensa a ineficiência inerente aos compiladores de alto nível. O sistema descrito aqui (SITASU) teve como pré-requisito no seu desenvolvimento a utilização de microcomputadores para garantir um baixo custo do equipamento; contudo, um microcomputador programado em alto nível não conseguiria executar as funções necessárias satisfazendo as limitações de tempo inerentes ao processo. Como resultado disto o software aplicativo do processo foi escrito em linguagem "assembly" do microprocessador 8085 (INTEL), utilizado como CPU do microcomputador controlador original.

Um outro agravante para uma mudança de CPU estava na disponibilidade de ferramentas para desenvolvimento de hardware e software, não só pelas dificuldades para a importação de equipamentos (prazos envolvidos) mas também pelos altos custos indiretos (taxas alfandegárias).

1.4.3 Arquitetura existente reformulada

As ferramentas disponíveis para a CPU de 8 bits utilizada até aquele momento (Z80), juntamente com a experiência adquirida com seu uso em outros projetos, quando comparadas aos recursos quase nulos disponíveis para uma CPU de 16 bits, criaram uma forte tendência para a manutenção da CPU original. Evidentemente, adicionando-se novas

características para a obtenção de maior desempenho. A seção seguinte traz mais detalhadas as idéias básicas desta reformulação.

1.5 Características adicionadas ao sistema

1.5.1 Características iniciais

O barramento padronizado (denominado via SME) é uma via de acesso paralela à qual se interfaceiam três tipos diferentes de placas, que o utilizam para realizar transferências de dados de 8 ou 16 bits. Estes três tipos são:

- placa tipo mestre (ex: placa de CPU), capaz de acionar sinais de controle da via de forma a iniciar transferências de dados.

- placa tipo escravo (ex: placas de memória, placas com dispositivos de E/S), capaz de responder a sinais de controle da via de forma a terminar transferências de dados iniciadas.

- placa tipo mestre/escravo (ex: placa de DMA), capaz de comandar transferências como uma placa tipo mestre e de ser comandada em transferências como uma placa tipo escravo.

1.5.2 Ciclos de acesso à via existentes

Os tipos de ciclos de acesso à via implementados no microcomputador básico e comandados por uma placa tipo mestre (CPU) eram (figura 1.1):

- ciclo de leitura de memória (T1).

- ciclo de escrita em memória (T2).

- ciclo de leitura de dispositivos de E/S (T3).

- ciclo de escrita em dispositivos de E/S (T4).

E os comandados por uma placa tipo mestre/escravo (DMA) eram:

- ciclo de leitura de memória e escrita em E/S simultâneos (T5).

- ciclo de leitura de E/S e escrita em memória simultâneos (T6).

1.5.3 Detalhes dos ciclos de DMA

Os ciclos de transferências através de DMA descritos acima são do tipo "flying-by", isto é, em apenas um ciclo realiza-se a transferência entre duas placas tipo escravo, sendo que o controlador de DMA não interage com o dado transferido, o qual percorre uma trajetória ao largo do

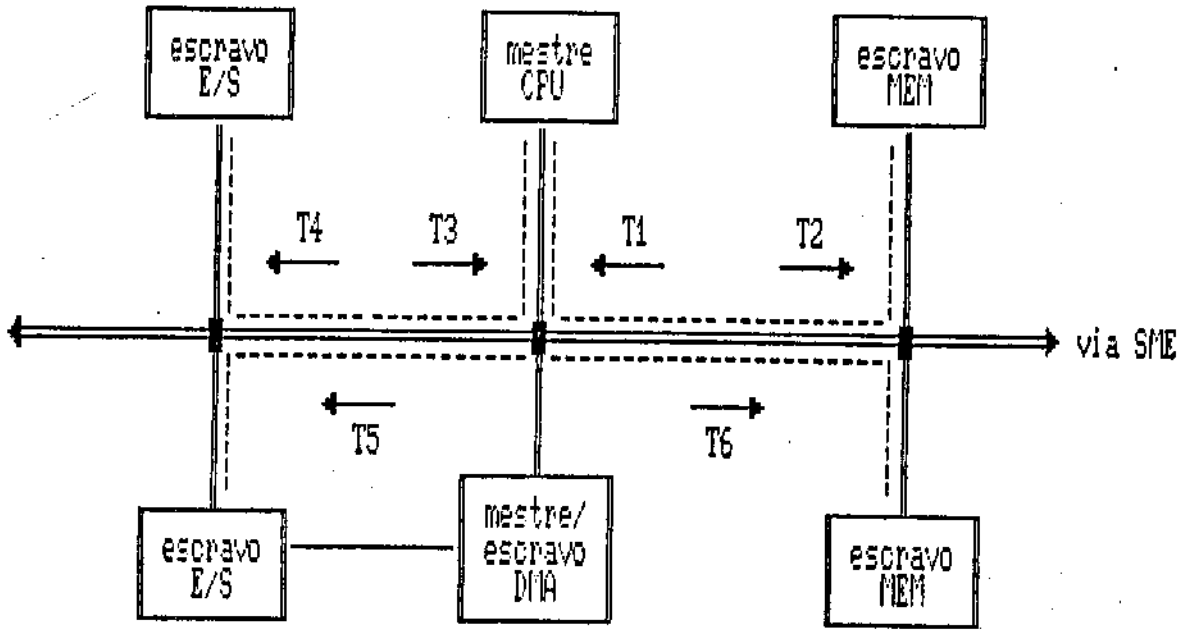


Figura 1.1 : tipos de transferencias atraves da via SME (implementacao inicial)

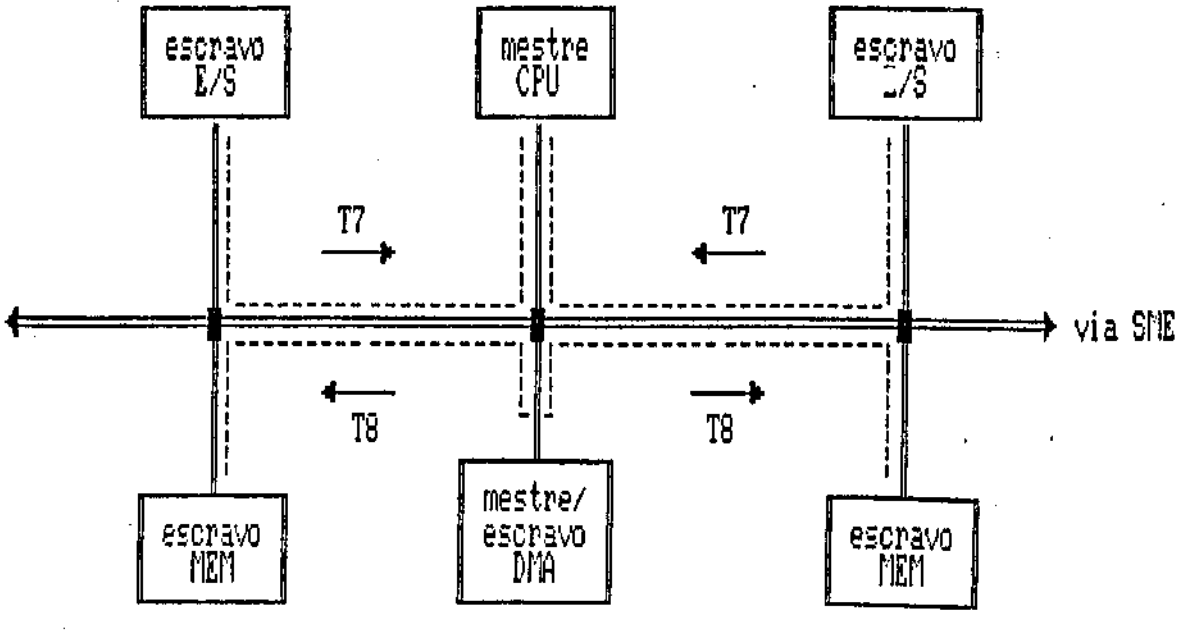


Figura 1.2 : tipos de transferencias adicionados a via SME.

controlador. A caracterização de um ciclo deste tipo é a seguinte:

sinais ativados por placa de DMA em ciclo tipo "flying-by"

acesso à memória	BE0-BE19	:	endereço
	/LEM	:	"strobe" para leitura de memória (ou /ESM - escrita em memória)
caracteriza transferência por DMA	/HLDA	:	CPU cedeu o barramento
acesso a E/S	/EES	:	"strobe" para escrita em E/S (ou /LES - leitura de E/S)
	/DACK	:	conexão especial da placa DMA com a placa de E/S em questão, substituindo as linhas de endereço que normalmente são usadas na seleção de uma placa de E/S

As placas de E/S decodificam 8 ou 9 bits de endereço quando o acesso é realizado por uma placa de CPU. Quando o comando do barramento é passado a uma placa de DMA (/HLDA ativo) o

decodificador interno de todas as placas de E/S é inibido. Somente aquela que receber um sinal /DACK ativo através de conexão especial vai responder ao ciclo iniciado. Desta forma, o dado lido da memória ou do dispositivo de E/S se faz presente no barramento de dados (após o tempo de acesso pertinente) e é escrito no dispositivo de E/S ou na memória, respectivamente, na desativação do "strobe" de escrita associado. O ciclo de DMA é suficientemente longo para atender aos requisitos de tempo de cada placa combinados.

1.5.4 Novos ciclos de acesso

Dois tipos de ciclos de acesso à via foram acrescentados (figura 1.2):

- ciclo de reconhecimento de interrupção com leitura de vetor fornecido pela placa que interrompe (T7), ciclo comandado por placa tipo mestre (CPU).

- ciclo de transferência memória à memória, na realidade um ciclo duplo constituído de um ciclo de leitura de memória seguido de um ciclo de escrita em memória (T8), comandado por placa tipo mestre/escravo (DMA). Este tipo de ciclo é chamado "flying-through", isto é, a transferência é realizada na realidade em dois ciclos indivisíveis, sendo que no primeiro o dado lido é armazenado temporariamente no controlador de DMA, para no segundo então ser escrito na placa de memória destino.

1.5.5 Localização das modificações de hardware

A localização das mudanças efetuadas no hardware é a seguinte:

- foi realizada uma redefinição de algumas linhas do barramento não plenamente utilizadas até então, de forma a implementar uma cadeia de prioridades para interrupções vetorizadas pelo barramento.

- na placa de CPU foi acrescentada a lógica necessária para a aquisição de vetores de interrupção, juntamente com um controlador de interrupções emulando este esquema para as placas que não dispõem de tal mecanismo. A disponibilidade de vários contadores, com a possibilidade de gerar interrupções independentes, fornece um bom suporte para executivos de tempo real e temporizações em geral.

- a placa controladora de DMA foi ampliada de 4 para 11 canais independentes, além de poder acessar o espaço total de endereçamento do microcomputador SME. Esta última característica também é independente para cada canal.

- a placa de interface série foi ampliada de 2 para 4 canais, operando com interrupções vetorizadas, interface para DMA disponível para os quatro canais, utilizando protocolos assíncronos e síncronos (BSC, SDLC) a velocidades de até 700 Kbps.

1.5.6 Objetivos procurados

O microprocessador Z80 endereça diretamente apenas 64 Kbytes de memória, sendo que as necessidades típicas previstas para os sistemas derivados do microcomputador SME iriam oscilar entre 128 e 384 Kbytes. O esquema de gerenciamento de memória que permitisse a utilização de memória física maior que a endereçável diretamente deveria também oferecer rapidez de operação e de manipulação para não provocar perdas significativas no desempenho global. O objetivo era fazer com que o executivo multitarefas suportasse o gerenciamento da aquisição de dados em tempo real, dos algoritmos de processamento, do armazenamento em memória de massa e das tarefas de comunicação.

O tratamento das interfaces de comunicação deveria ser aperfeiçoado a fim de evitar laços de software realizando "polling" do estado destas interfaces. Deve-se levar em conta que certas aplicações poderiam exigir cerca de 10 ou mais canais de comunicação série. O esquema de interrupções vetorizadas, somado aos "chips" de interface Z80-SIO fazendo pleno uso desta facilidade, aliviariam bastante a carga de processamento da CPU em favor de outras tarefas. Para o caso de transferência de dados em blocos de informação, uma outra facilidade disponível seria a utilização de DMA, proporcionando muito mais tempo livre para a CPU quando comparada à utilização de interrupção.

Dependendo da configuração do sistema, os dispositivos que poderiam exigir esta facilidade não seriam poucos:

controlador de diskette, aquisição de dados de processos, congelamento de blocos de dados na memória (através de transferências de DMA tipo memória à memória), e possivelmente vários canais de interface série. O projeto tradicional de uma lógica de controle de DMA utilizaria um "chip" LSI (por exemplo o modelo 8237 da INTEL), mas seus 4 canais seriam obviamente insuficientes, como também seu reduzido espaço de endereçamento de 64 Kbytes. Dotou-se pois a placa de 11 canais, um dos quais capaz de realizar transferências memória à memória, sendo cada um deles com registro de segmentação próprio para o acesso completo aos 1 Mbyte do microcomputador.

A maneira como foram implementadas estas modificações será detalhada nos capítulos seguintes.

CAPÍTULO 2

UNIDADE DE GERENCIAMENTO DE MEMÓRIA

Este capítulo analisa alternativas para se obter um adequado gerenciamento de memória, seguindo com o desenvolvimento do esquema adotado e discutindo as características do hardware obtido.

2.1 Objetivo primário: ampliação do espaço de endereçamento

O problema básico apresentado consiste no mapeamento de um pequeno espaço lógico de endereçamento em um espaço físico maior de memória.

A CPU Z80 gera endereços lógicos de 16 bits (espaço total de 64 Kbytes) que devem ser mapeados em endereços físicos de 20 bits (espaço total de 1 Mbyte) do microcomputador SME.

Um sistema de gerenciamento de memória, por mais elaborado que seja, não eliminará os problemas decorrentes do limitado espaço linear de endereçamento, imposição ditada pela utilização de uma CPU em VLSI, sem possibilidade de alterações em sua arquitetura. A solução adotada deve portanto tratar da minimização destes problemas e da sobrecarga no processamento da CPU, em conjunto com a adição de facilidades extras, embutidas na solução, tentando compensar as desvantagens.

2.2 Alternativas para gerenciamento

2.2.1 Segmentação via seleção de bancos

A maneira mais fácil de se aumentar a capacidade de endereçamento de uma CPU é adicionar-se um registro cujo conteúdo seja utilizado para criar linhas de endereço extras (figura 2.1). A simplicidade do hardware apresentado esconde porém a necessidade de algumas adaptações. A seleção de um novo banco de memória (alteração no conteúdo do registro de seleção) pode ser catastrófica, com a truncagem do programa em execução na CPU. A elaboração de um esquema coerente de entradas e saídas dos segmentos seria confusa, assim como sua utilização pela CPU. A solução usual neste caso é a manutenção de um pequeno espaço de memória sempre ativo para qualquer segmento selecionado, criando uma base estável para o programa de comutação de segmentos. A desvantagem está na subutilização dos bancos de memória. A figura 2.2 exemplifica uma memória sempre ativa de 8 Kbytes, abrindo portanto janelas de 8 Kbytes na decodificação dos bancos de memória de 64 Kbytes, as quais não podem ser utilizadas (figura 2.3).

O cálculo do endereço físico pode ser expresso pela seguinte fórmula:

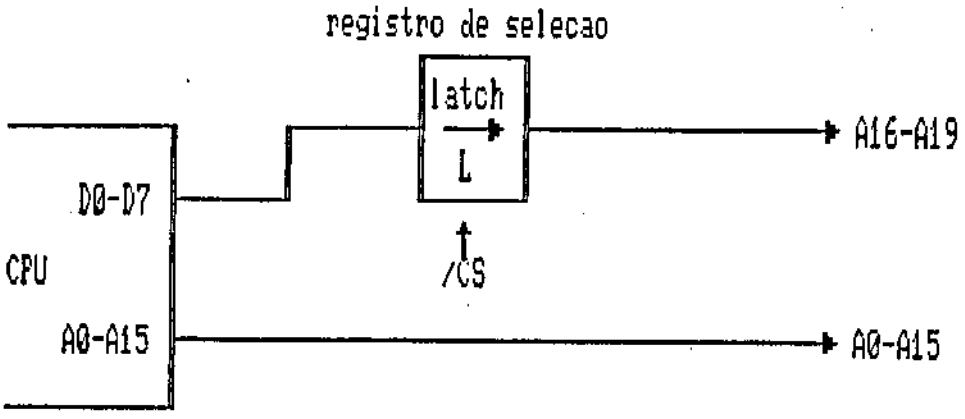


Figura 2.1 : segmentacao (selecao de bancos)

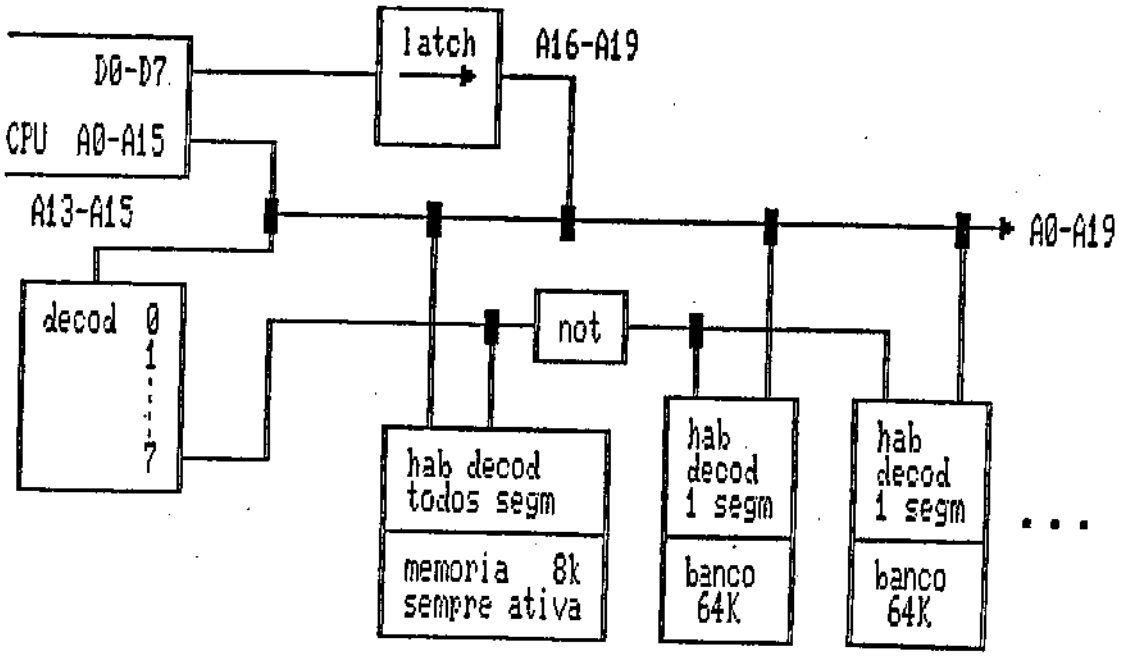


Figura 2.2 : base comum para comutacao de segmentos

EF = --> BANSEL * 2**16 + EL para 0 <= EL <= DFFFH

--> 0 + EL para E000H <= EL <= FFFFH

onde: EL = endereço lógico
EF = endereço físico
BANSEL = número do banco selecionado

obs: para espaço físico de 1 Mbyte, 0 <= BANSEL <= FH

2.2.2 Segmentação via mapeadores de segmentos

Permitem dividir o espaço lógico de endereçamento em um número fixo de segmentos lógicos, de tamanho fixo ou variável, e associá-los a segmentos físicos (na memória) de mesmo tamanho. Segmentos de tamanho variável são desejáveis, pois minimizam o número de registros a serem utilizados no mapeamento (ver item "Paginação", seção 2.2.4). A figura 2.4 detalha o funcionamento deste esquema [SPRY81]. As implementações com segmentos de tamanho variável geralmente impõem a restrição de que o tamanho de cada segmento, assim como seus limites, seja múltiplo de 2 [MOTO81], para que se tenha grande simplificação na lógica envolvida.

A principal desvantagem deste esquema está na subutilização de memória, fato que pode ocorrer freqüentemente. Se os segmentos tiverem tamanho variável esta subutilização pode ser contornada, mas somente até certo ponto, como veremos a seguir.

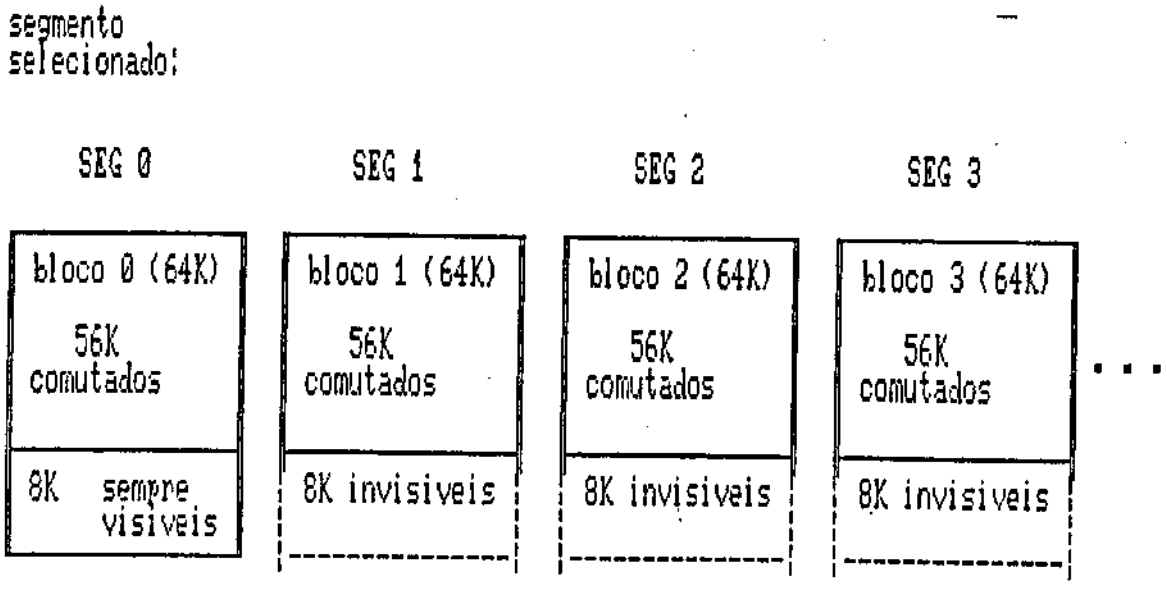


Figura 2.3 : mapeamento de memoria em um esquema de segmentacao (selecao de bancos)

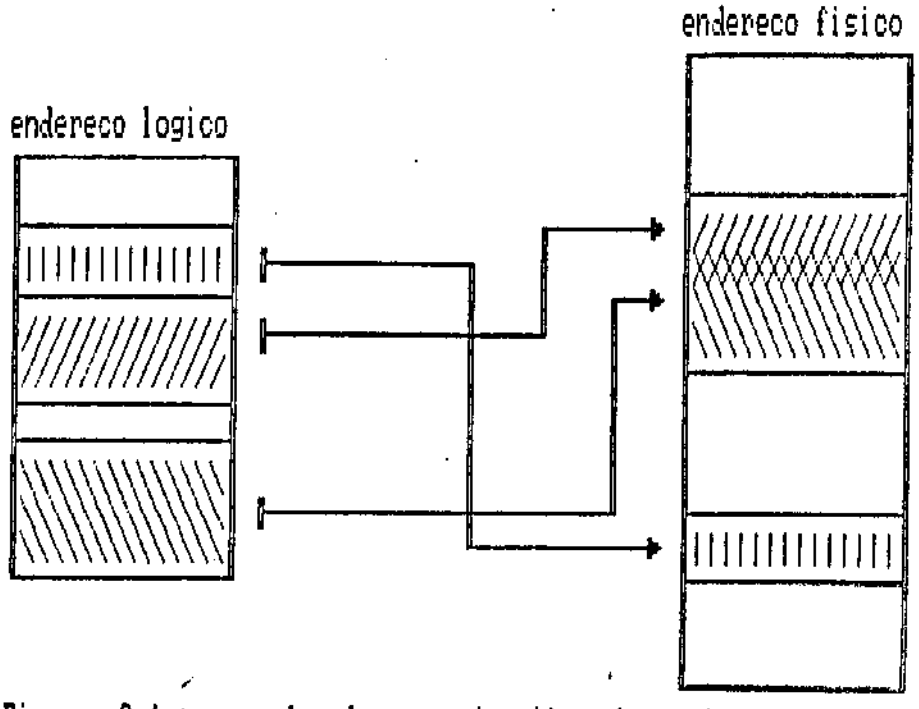


Figura 2.4 : mapeador de segmentos (tamanho variavel)

Imaginemos o caso de um processo cuja área de dados ocupe 17 Kbytes. Se para o mesmo for alocado um segmento de 32 Kbytes, o espaço restante (15 Kbytes) não poderá ser utilizado por outro processo, exceto se houver sobreposição com um outro segmento. Como isto geralmente não é desejável, podem ser alocados dois segmentos de tamanhos diferentes, um de 16 Kbytes e um de 1 Kbyte. A utilização integral por outra tarefa dos 15 Kbytes restantes exigirá 4 segmentos (8K, 4K, 2K e 1K). Esta utilização de descritores múltiplos (caracterizando um contexto) é restringida pelo número finito de descritores que podem residir estaticamente em uma unidade de gerenciamento típica, como é o caso do 68451 da Motorola [MOTO81], que pode armazenar até 32 descritores. A quantidade de contextos residentes será bastante limitada, o que exigirá, para um número maior de processos, freqüentes recarregamentos dos descritores na unidade de gerenciamento, causando uma perda significativa na capacidade de processamento da CPU.

O cálculo do endereço físico neste caso pode ser expresso assim:

EF = MAPSEG(EL) * 2**k + EL mod 2**k

onde: EL = endereço lógico

EF = endereço físico

MAPSEG(EL) = função que mapeia um segmento lógico em um segmento físico

k = número de bits (menos significativos) não abrangidos pela função de mapeamento (definindo o tamanho do segmento)

2.2.3 Segmentos variáveis contíguos

Um esquema de gerenciamento muito interessante é apresentado em [BRON82], no qual os segmentos têm tamanhos variáveis (não se limitando às potências inteiras de 2), e onde o espaço lógico completo de endereçamento da CPU é dividido em um número fixo de segmentos lógicos, todos contíguos. A tradução de endereços é realizada via somadores, e não através de uma máscara que permite atuação somente nos bits mais significativos, como é o caso dos mapeadores de segmentos. Uma maior flexibilidade é obtida quando se aumenta o número de segmentos com que é feita a partição do espaço lógico.

O endereço físico é obtido do seguinte modo:

- dado que o espaço lógico vai de 0 a M-1, é feita uma partição

  A0 = 0 < A1 < A2 < ... < Ai < ... < Al = M

- a cada registro Ai corresponde um registro Bi

- desde que Ai <= EL < Ai+1, temos:

- EF = Bi + (EL - Ai)

  onde: EL = endereço lógico

  EF = endereço físico

  Ai = fronteira lógica de um segmento

  Bi = deslocamento realizado nos endereços de um segmento

Uma simplificação importante consiste em retirar alguns dos bits menos significativos da lógica, digamos k:

- desde que

  Ai * 2**k <= EL < Ai+1 * 2**k

  temos:

- EF = Bi * 2**k + (EL - Ai * 2**k)

A utilização deste esquema no microcomputador objeto do estudo seria fisicamente inviável em vista do tamanho de suas placas. Note-se que as lógicas de comparação e adição devem ser replicadas ao número de segmentos da partição escolhida, para que o atraso total de tradução de endereço não obrigue à inclusão de "wait-states" nos ciclos de acesso

da CPU.

2.2.4 Paginação

Neste esquema o espaço lógico de endereçamento da CPU é completamente dividido em segmentos de tamanhos fixos e iguais, chamados páginas lógicas [GUIM79] [NATI84]. A cada página lógica corresponde uma página física na memória, de mesmo tamanho. Uma simplificação evidente na lógica restringe as fronteiras das páginas físicas a múltiplos de 2 ** k, onde 2 ** k é o tamanho da página lógica. Isto significa que os k bits menos significativos não são traduzidos, mas compõem diretamente o endereço físico. A figura 2.5 detalha o funcionamento do esquema de paginação. Na figura 2.6 pode-se observar a trajetória efetuada pelas linhas de endereço até a composição do endereço físico, mostrando claramente a simplificação na lógica quanto aos k bits menos significativos.

A estrutura de software para os novos sistemas UCR e USCE, já parcialmente definida, permitia esboçar as configurações de memória necessárias para as diversas tarefas dos sistemas. A flexibilidade na definição do tamanho dos segmentos num esquema de segmentos variáveis contíguos seria muito bem utilizada nestes sistemas, permitindo uma otimização na utilização do exíguo espaço lógico disponível. A implementação deste esquema não seria viável, devido ao espaço físico necessário para os circuitos envolvidos. Em vista disso a solução adotada foi o esquema de paginação, principalmente pelas vantagens extras obtidas (grande número

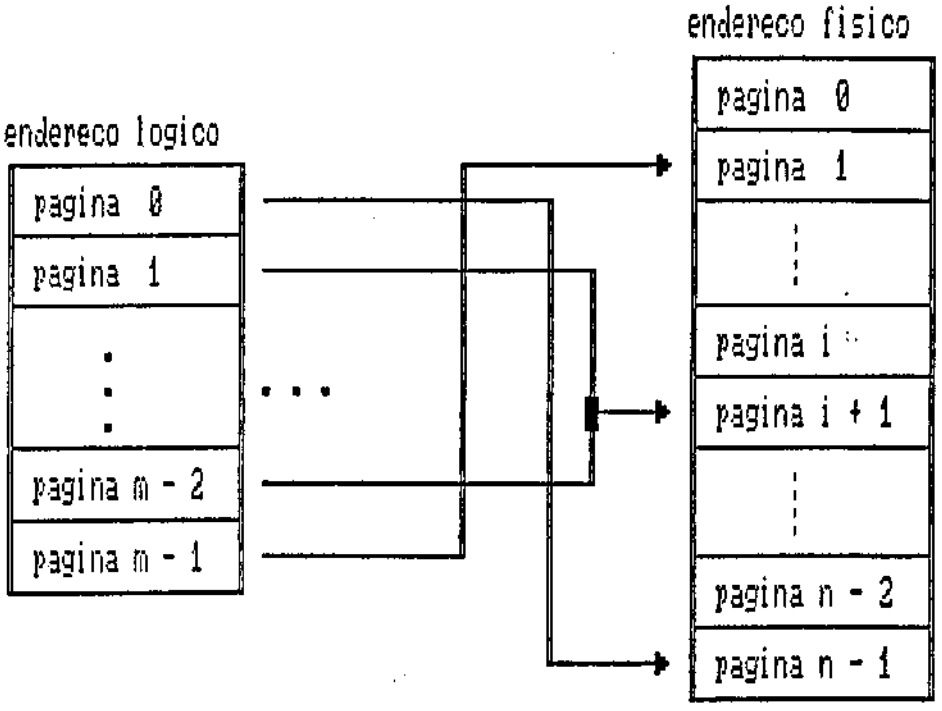


Figura 2.5 : esquema de paginacao.

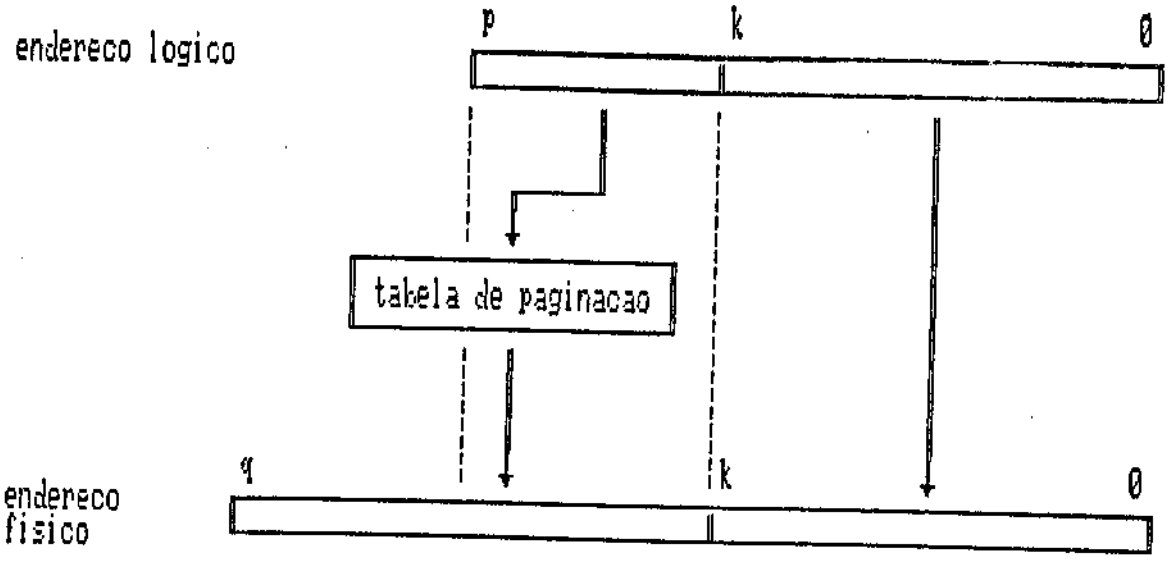


Figura 2.6 : rastreamento das vias de endereco na composicao do endereco fisico, esquema de paginacao.

de contextos residindo estaticamente, seção 2.3.2) e pelas facilidades proporcionadas pelo hardware da CPU Z80, como veremos a seguir.

2.3 Detalhamento da unidade de gerenciamento de memória (MMU)

2.3.1 Obtenção do hardware básico

Os tamanhos de páginas encontrados na maioria dos sistemas que utilizam este esquema oscilam entre 256 bytes e 4 Kbytes. O tamanho de página a ser adotado deveria ser uma solução de compromisso entre a flexibilidade desejada (páginas menores) e a simplificação ou viabilidade da lógica envolvida (páginas maiores).

Tendo como referência a figura 2.6, o acesso à tabela de paginação foi mapeado no espaço de E/S da CPU Z80, para garantir livre o espaço total de endereçamento de memória, já demasiado restrito. Quando o dispositivo está em operação os bits superiores do endereço gerado pela CPU são utilizados como índice dentro da tabela de paginação, selecionando uma de suas entradas para fornecer os bits superiores do endereço físico que vai à memória do sistema. As linhas de endereço que serviram como índice na tabela serão traduzidas, ao mesmo tempo que novas linhas serão criadas, permitindo o acesso a um espaço de endereçamento maior que o da CPU.

Para se modificar o conteúdo dos registradores desta tabela, seria necessário um "latch" (previamente carregado) servindo como índice na tabela, ao passo que o endereço normal de E/S (de 8 bits) seria decodificado para habilitar um "transceiver" conectando as entradas de dados da tabela às linhas de dados da CPU. No entanto, uma nova instrução adicionada à CPU Z80 em relação à CPU 8085 torna dispensável esta lógica [DILL82]. A instrução OUT (C),r permite uma escrita de E/S com 16 bits de endereço: os 8 bits inferiores serão o conteúdo do registro C, os 8 bits superiores o conteúdo do registro B, e o conteúdo de r, por exemplo o registro A, será colocado nas linhas de dados. Portanto, variando-se B obtém-se acesso a todas as entradas da tabela, desde que se defina k e p (figura 2.6):

$$k + 1 >= 8 \quad , \text{ com}$$

$$p = 15 \quad .$$

Observa-se ainda que para manter a lógica simples devemos ter

$$q - k <= 8 \quad ,$$

ou seja, apenas um "transceiver" para cada registrador. Além disso,

q - p = 4 , pois

q = 19 (1 Mbyte) e

p = 15 (Z80).

Desta forma, o requisito de maior flexibilidade, sob as restrições anteriores, nos leva a

(4 + p) - k = 8 , e portanto:

k = p - 4 = 11

Como resultado obtemos uma tabela de 16 entradas (p - k = 4), portanto 16 páginas, de tamanho 2 ** (k + 1), ou seja, 4 Kbytes.

Esta tabela de paginação poderia ser implementada através de 16 "latches" de 8 bits, com o auxílio de um decodificador 4 para 16. Ou, de forma condensada, por uma memória RAM em configuração 16 x 8 , bastante rápida, com tempo de acesso da ordem de dezenas de nanossegundos (ver seção 2.3.3) para não prejudicar o tempo de acesso à memória física.

2.3.2 Contextos residentes

Definida a estrutura básica, aparece um problema decorrente da flexibilidade desejada. A comutação para uma outra tarefa poderá exigir a atualização de até 15 entradas da tabela de paginação, ou mesmo 16, em certas situações.

Examinando-se o funcionamento dinâmico do sistema, observa-se que comutações de tarefas ocorrerão não somente pelas requisições do relógio de tarefas do sistema, mas também quando da ocorrência das demais interrupções do sistema (canais de comunicação série, sinalização dos controladores de DMA e de memória de massa). Naturalmente estas últimas necessidades de comutação serão ditadas pela possibilidade ou não de as rotinas de atendimento residirem permanentemente dentro do reduzido espaço de endereçamento da CPU, juntamente com a tarefa temporária presentemente ativa. Resumindo, a perda de desempenho ocasionada pela necessidade de constantes carregamentos da tabela de paginação pode alcançar valores significativos.

Em máquinas grandes a implementação de um esquema de paginação quase sempre envolve o uso de memória tipo "cache" para a obtenção de um endereço traduzido, valendo-se de algoritmos eficientes de atualização tais como FIFO (o primeiro que entra é o primeiro que sai) ou LRU (o menos recentemente usado) [SMIT82]. A razão para isto pode ser exemplificada para o caso da CPU 32016 e a MMU associada 32082. As páginas aqui são de 512 bytes, ou seja, (figura 2.6)

$$k = 8 \quad .$$

Como o espaço lógico é de 16 Mbytes temos

$$p = 23 \quad ,$$

o que exigiria uma memória de 32K x 15 bits para alojar apenas um contexto (supondo espaço físico também de 16 Mbytes). Ao invés disso, a memória da MMU é de apenas 32 x 15 bits, cujo algoritmo de atualização procura utilizá-la da maneira mais inteligente possível, dispondo de um hardware complexo dedicado. A perda de processamento ocasionada por estas atualizações será proporcional à quantidade de memória acessada pela tarefa. A adoção de tal mecanismo não seria viável em nosso caso, devido às proporções físicas que o hardware envolvido iria tomar; isso fez com que fosse buscada uma solução baseada no esquema obtido anteriormente. A solução proposta é manter várias tabelas carregadas ao invés de apenas uma, comutando-se de tabela através de uma única operação, gerando uma perda de processamento bastante baixa.

O primeiro requisito para a memória que implementa as tabelas de paginação é a rapidez, o que até recentemente obrigava a utilização de tecnologia bipolar (TTL ou ECL) nestes dispositivos. A evolução dos processos de fabricação tornou disponível dispositivos de memória de tecnologia NMOS tão rápidos quanto de tecnologia TTL, com dissipação de energia bem menor, e consequentemente com maiores possibilidades em termos de integração. O dispositivo selecionado para este projeto foi a 2148, RAM 1K x 4 bits, do qual se encontra atualmente versões de até 25 ns. O dispositivo 6148, tornado disponível recentemente, é funcionalmente idêntico e tão rápido quanto, mas de tecnologia CMOS e dissipando apenas 20 % da potência da 2148.

Com a utilização de dois destes dispositivos obteve-se uma memória de 1K x 8 bits, onde a formação por 8 satisfaz a solução adotada

$$q - k = 8 \quad .$$

Por outro lado, como

$$p - k = 4 \quad ,$$

nota-se que dos 10 bits de endereço desta memória, 6 estão disponíveis. Alimentando estas linhas foi adicionado um "latch" de 6 bits, chamado de registro de contexto. Tem-se então 2 ** 6 = 64 contextos diferentes, cada um com uma tabela de 16 páginas de 4 Kbytes. O esquema definitivo pode ser visto na figura 2.7 [SPRY81] e na figura 2.9 estão esquematizados os circuitos de hardware.

2.3.3 Análise do "timing" envolvido

Para análise dos tempos envolvidos no funcionamento do circuito examinaremos o caso mais crítico para a CPU Z80, que é o ciclo de busca de código de instrução ("op-code fetch"). Os dados fornecidos no formato Tx / Ty relacionam-se às freqüências 4,096MHz / 6,144MHz para o "clock" da CPU Z80. Note-se que os atrasos dependentes do clock devem ser aproximadamente proporcionais, ou seja, se por exemplo Tend se apresentar em um mínimo, o mesmo deve acontecer com t4. Fatores predominantes sobre estes atrasos são a tensão de alimentação, a carga nos "buffers" de saída

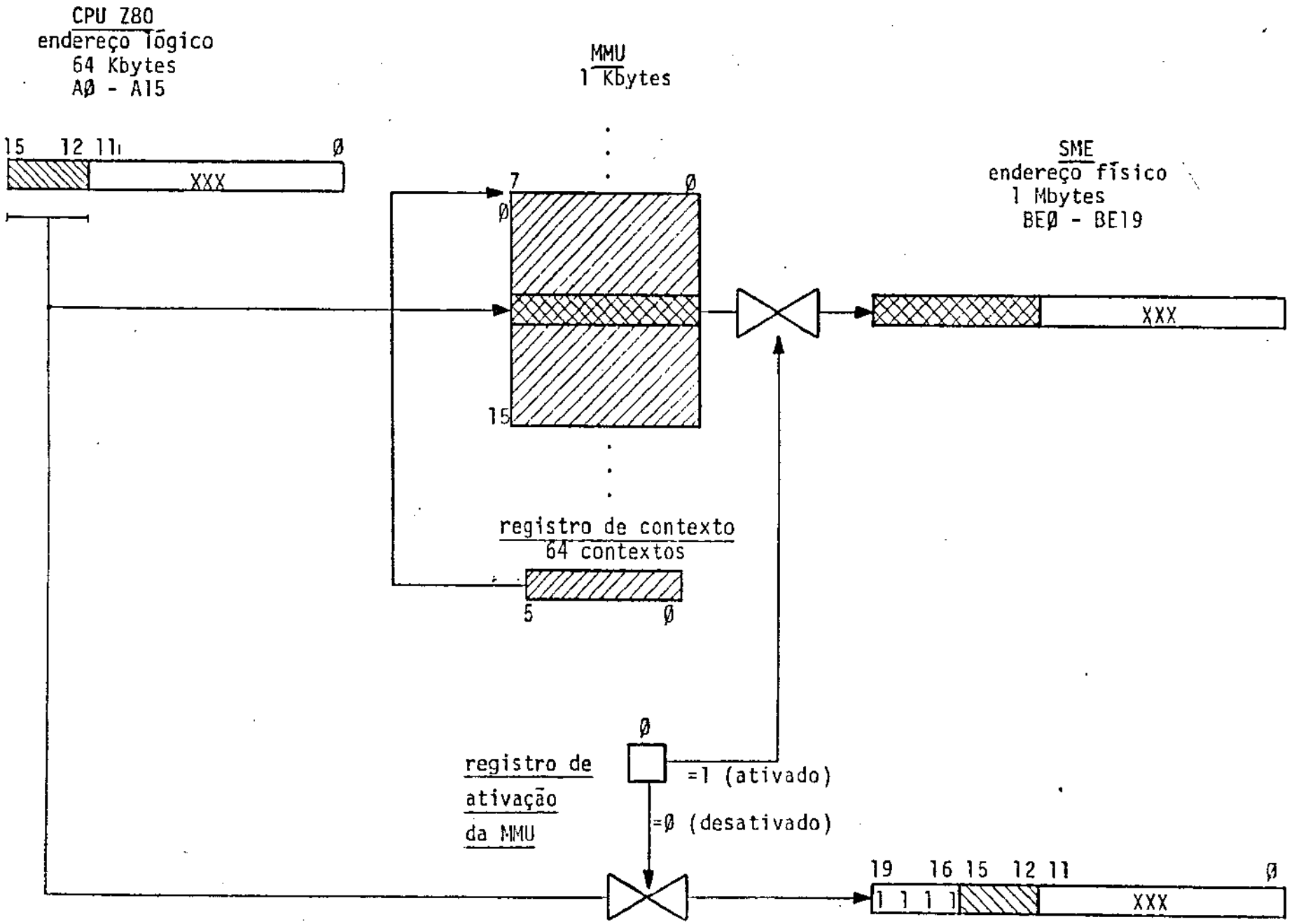


Figura 2.7 : diagrama de funcionamento da MMU

e a temperatura ambiente, agindo de forma eqüitativa sobre os "buffers" de saída dos sinais. O diagrama de tempo associado é dado na figura 2.8.

A utilização de memória estática para a tabela de paginação permite que o acesso nesta memória especial seja iniciado imediatamente após o tempo de estabilização das linhas de endereço da CPU, no início do ciclo. Para se obter este resultado coloca-se a memória em um estado de leitura permanente, isto é, o sinal /CE permanentemente ativado e o sinal /WE permanentemente desativado (nota: /WE é ativado quando dados são carregados na MMU). Evita-se assim a criação de um sinal síncrono para iniciar a leitura, o que certamente adicionaria algum atraso indesejável. O protocolo de comunicação da via SME impõe que as linhas de endereço devam estar estáveis antes que os "strobes" de leitura/escrita sejam ativados. Pela figura 2.8 devemos ter:

$$Tend + Tmmu < Tread + Tclk/2$$

onde:

Tend é o atraso de estabilização das linhas de endereço
= 145/110 ns (máx)

Tmmu é o tempo de acesso da memória da MMU
= 35, 45, 55 ns (máx)

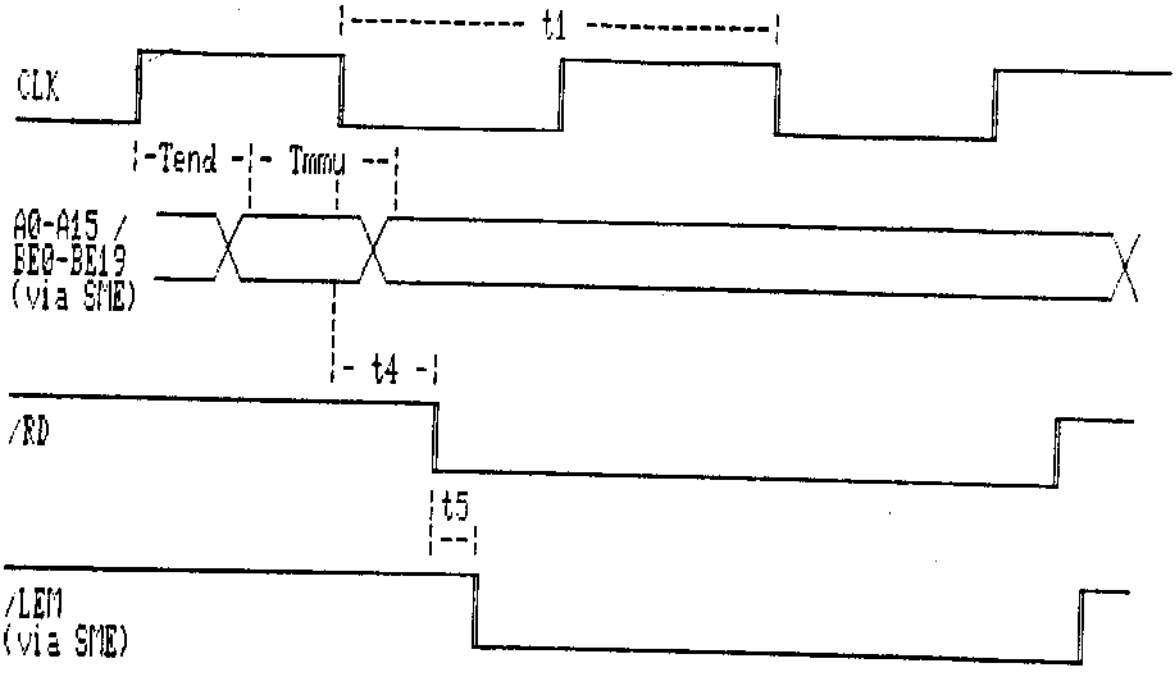


Figura 2.8 : ciclo de busca de codigo de instrucao da CPU Z80

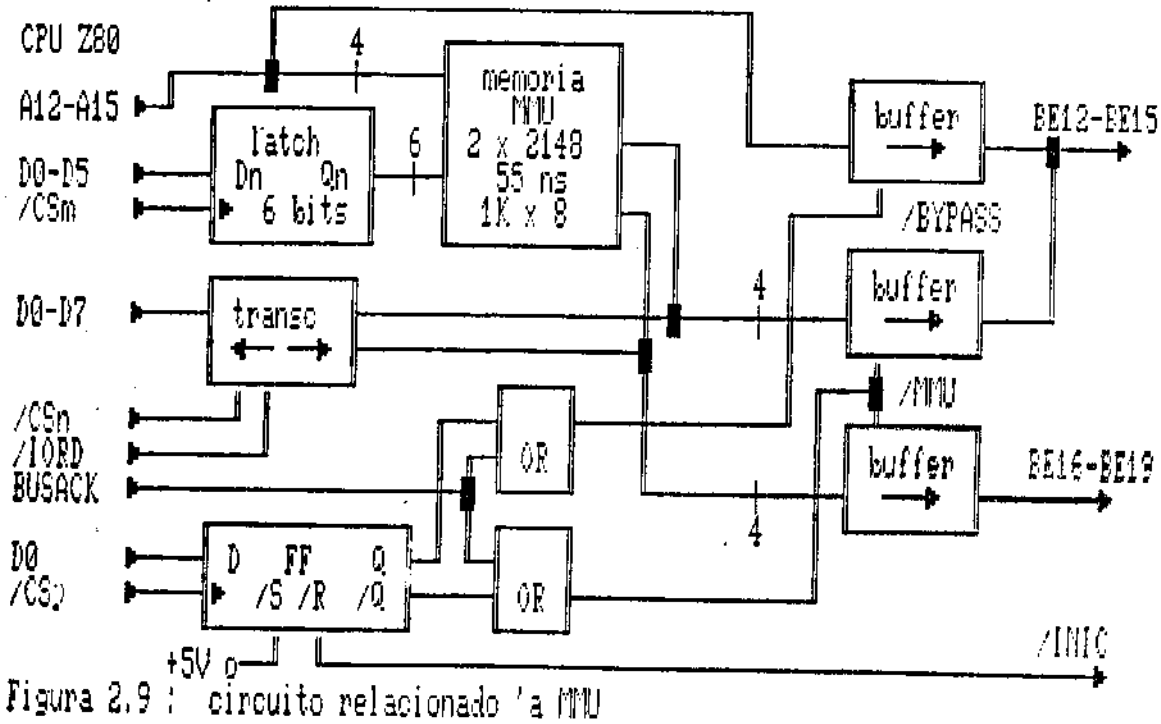


Figura 2.9 : circuito relacionado 'a MMU

Tread = t4 máx + t5 mín ; atraso do "strobe" de leitura

t4 = 130/95 ns (máx) ; (definido pelo fabricante)

t5 = 5 ns (mín) ; (atraso na geração do sinal /LEM)

ou

Tread = 135/100 ns

tem-se também:

Tclk = t1 = 244/163 ns

Tclk/2 = 122/81 ns

Obtemos então:

Tmmu < Tread + Tclk/2 - Tend

< 135/100 + 122/81 - 145/110

ou seja,

Tmmu < 112/71 ns .

Vemos portanto que qualquer das velocidades citadas para a memória da MMU satisfaz os requisitos básicos da via.

2.3.4 Estado após inicialização

Após a inicialização do sistema a MMU está desativada, e o endereço físico presente no barramento será idêntico ao endereço lógico da CPU Z80 (figura 2.7), a menos das linhas BE16-BE19, as quais não são geradas diretamente pela CPU. Na realidade estes sinais não são acionados nesta situação, com seus "buffers" colocados em estado de alta impedância; a terminação resistiva do barramento define-os como "1", e o registro de ativação da MMU em estado "0" (desativada) força o acionamento do barramento de endereços através da trajetória inferior (figura 2.7). O registro de ativação da MMU é acessado através do bit 0 do acumulador, mapeado no espaço de E/S da mesma forma que o registro de contexto e o "transceiver" de acesso aos registros da tabela de paginação (seção 2.3.1).

Se considerarmos o espaço de 1 Mbyte do SME como 16 segmentos contíguos de 64 Kbytes, ordenados de "0" a "F", podemos dizer que após a inicialização, ou enquanto a MMU estiver desativada, será acessado somente o segmento "F", ou seja, o espaço físico correspondente aos endereços F0000H a FFFFFH da via SME. A posição F0000H deve ser ocupada pelo início do software de partida e/ou "bootstrap" da máquina.

2.3.5 Operação da MMU

Antes de se ativar a MMU, o contexto desejado deve ser carregado; todas as entradas da tabela desejada devem refletir o mapeamento de memória desejado. Uma limitação

imposta pela simplificação da lógica impede o carregamento de um contexto quando um outro está ativo. Esta operação deve portanto ser feita com a MMU desativada.

De forma genérica, as rotinas do executivo utilizadas para manipulação da MMU devem estar localizadas em uma mesma página lógica e física, devendo estar presentes em todos os diferentes contextos utilizados, juntamente com o núcleo do executivo. Imaginemos dois contextos previamente inicializados e uma rotina sendo executada no primeiro que irá selecionar o segundo. Digamos que a instrução OUTPUT selecionando o outro contexto esteja localizada logicamente nos endereços LXXXH e LXXXH+1, e fisicamente nos endereços PPXXXH e PPXXXH+1. O objetivo desta mudança seria permitir a chamada do ponto de entrada de uma nova tarefa a ser executada, utilizando uma rotina que se inicia em LXXXH+2 / PPXXXH+2. Porém, para que esta rotina seja executada imediatamente após a instrução OUTPUT é necessário que, na nova tabela selecionada, o registro da MMU correspondente à página lógica "L" forneça uma saída também igual a "PP". Se a saída fornecida "QQ" for diferente de "PP", a próxima instrução a ser executada estará no endereço QQXXXH+2, e não em PPXXXH+2.

As operações de ativação e desativação da MMU e de mudança de contexto têm a duração de 11 ciclos do "clock" da CPU Z80, ou 2,7 us a 4 MHz. As chances de que o software básico e o aplicativo do sistema ocupem mais do que 64 contextos são remotas, nas suas possíveis aplicações em controle e supervisão de processos. Isto permite supor que os diversos

contextos poderão residir estaticamente na MMU.

Considerando-se agora a situação após a inicialização de todos os contextos da aplicação, nota-se que na maioria das vezes deve ser feita apenas uma mudança de contexto para cada fatia de tempo alocada a uma tarefa pelo executivo. Somente as tarefas que utilizam muita memória (por exemplo, manipulando uma grande estrutura de dados) exigirão mudanças mais freqüentes. Os baixos tempos de comutação do hardware, combinados à pequena taxa de mudanças garantida pelo esquema, fazem com que as rotinas de chaveamento de contexto sejam executadas o menos possível, implicando em menor perda de processamento. Observe-se que o tempo de chaveamento de tarefas será determinado quase que somente pelo executivo, pois o tratamento do hardware exige uma pequena rotina com passagem de apenas um parâmetro (contexto desejado).

CAPÍTULO 3

INTERRUPÇÕES VETORIZADAS

Este capítulo apresenta o esquema de interrupções existente no sistema inicial e o da família Z80, além de outros semelhantes. Apresenta o funcionamento do esquema implementado, as modificações no hardware e as facilidades obtidas

3.1 Sistema de interrupções para tempo real

Um sistema para processamento em tempo real deve possuir características especiais para tratamento de interrupções. Prasser [PRAS70] relaciona os requisitos necessários para obtê-las, agrupados a seguir conforme as características para as quais contribuem:

- característica de tempo de reconhecimento curto (figura 3.1):

  - identificação automática dos caminhos a seguir quando da ocorrência de interrupções.

  - suporte de hardware para salvar o status da CPU quando dos desvios de programa causados por interrupções (atualmente as CPU's em LSI já trazem este mecanismo incorporado).

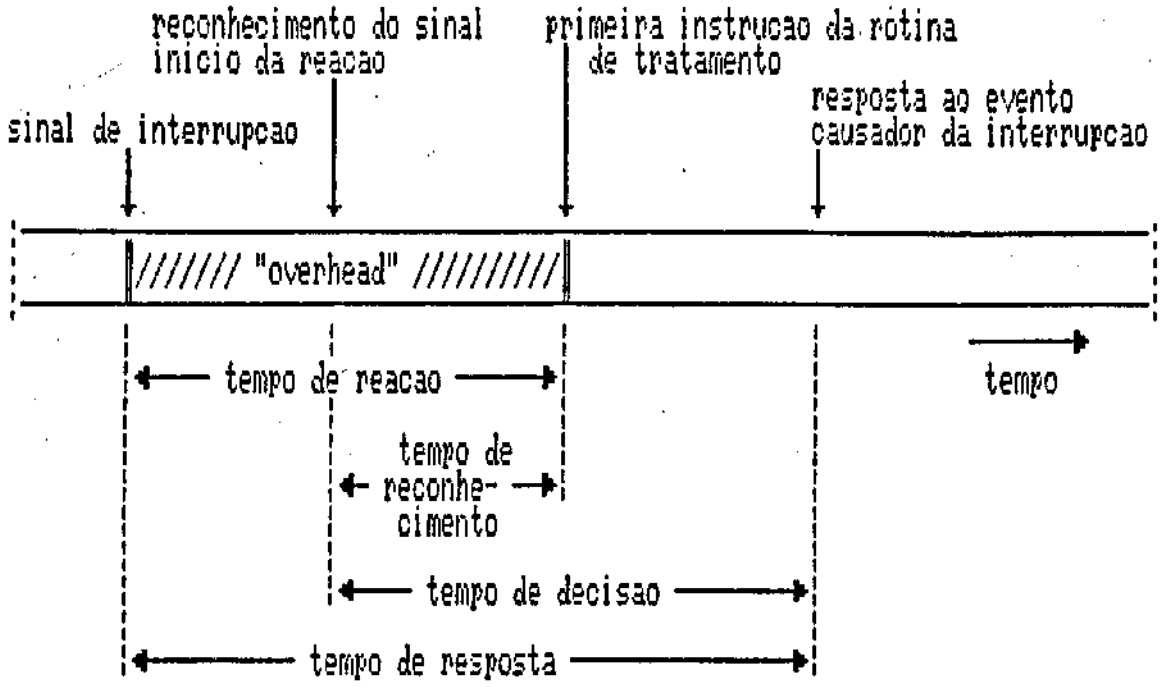


Figura 3.1 : definicao de tempos

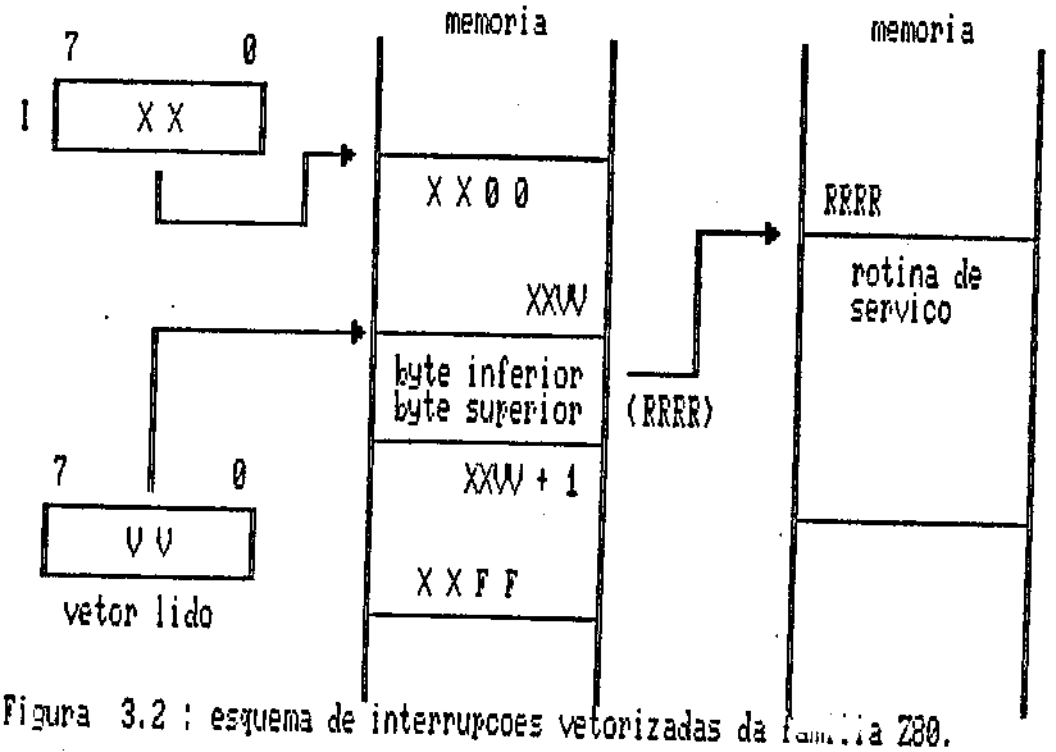


Figura 3.2 : esquema de interrupcoes vetorizadas da [illegible] Z80.

- característica de tempo de resposta curto:

    - interruptibilidade frequente (tarefas demoradas devem ser interrompíveis).

    - possibilidade de interrupções aninhadas (tempo de resposta mais curto quanto mais alta a prioridade).

    - flexibilidade nas rotinas de interrupção.

- característica de controle dinâmico de prioridades por programa:

    - controle de interrupções com desabilitação geral e máscaras (comandos eficientes para o controle de interrupções).

    - possibilidade de utilizar comandos capazes de simular situações de interrupção.

    - possibilidade de ordenação de prioridades.

- característica de confiabilidade:

    - reconhecimento diferenciado de erros, alta prioridade para sinais de erro.

O esquema implementado, descrito a seguir, satisfaz quase todos estes requisitos. A característica de controle dinâmico de prioridades não está completamente disponível,

pois a ordenação de prioridades é estática, fixada por hardware. Alguns dos requisitos são atendidos por projeto de software, e os restantes pelo hardware.

3.2 Esquema inicial

Como foi visto na seção 1.2, a via SME dispõe de 8 linhas de interrupção. A primeira placa de CPU desenvolvida para este sistema utilizava um controlador de interrupções da linha INTEL (8259), possibilitando a existência de 8 níveis de interrupção. A concepção geral do sistema, começando pela definição do barramento e do tamanho das placas, foi orteada para a descentralização de funções, de modo a permitir uma fácil expansão do sistema e a conseguir a modularidade desejada para suas aplicações. A necessidade de um número maior de níveis seria coberta pela utilização de outro "chip" deste modelo trabalhando em modo escravo, localizado na placa periférica que requisita estes níveis extras. A inclusão desta lógica poderia em certos casos inviabilizar a implementação física de certas placas funcionais, devido ao espaço físico necessário para o controlador e lógicas de apoio.

3.3 Esquema implementado

3.3.1 Esquema Z80 na via SME

A utilização de "chips" da família Z80 possibilitou a implementação de um esquema de interrupções vetorizadas sem a inclusão de lógica adicional, a qual é fornecida pelos próprios "chips" LSI da família, porém nem sempre a função desejada poderia ser implementada utilizando-se "chips" desta linha. Além disso, muitas placas funcionais já estavam implementadas a esta época.

Previu-se que a maior parte dos níveis de interrupção seriam requeridos por placas de interface série; sendo assim, foi escolhido o chip Z80-SIO para a nova implementação deste tipo de placa. 7 das 8 linhas de interrupção do barramento seriam reservadas às outras placas funcionais, desprovidas desta característica, que seria emulada por dispositivos localizados na placa de CPU. A linha restante seria então compartilhada pelas placas capazes de gerar vetores de interrupção. Associada a esta linha foi implementada uma cadeia de prioridades, utilizando dois pinos do conector de barramento. Durante ciclos de reconhecimento de interrupção esta cadeia, através de lógica associada à mesma em cada placa, habilitará apenas uma placa a colocar seu vetor no barramento de dados (ver seção 3.3.3).

Desta forma a limitação inicial de apenas 8 interrupções diferentes é expandida para 128 vetores. Mais do que isso: uma melhor utilização da capacidade de vetorização, através

da geração de vetores diferentes para categorias distintas de eventos da função de uma placa, pode agilizar as rotinas de serviço, evitando análise de status para identificação do correto tratamento a ser dado ao evento.

3.3.2 Funcionamento do esquema Z80 e semelhantes

Este esquema supõe que a CPU do sistema mantenha um apontador para uma tabela de vetores. No caso da CPU Z80 é o registro I [ZILO81], que localiza a tabela dentro do espaço de endereçamento da CPU (64 Kbytes), como é visto na figura 3.2. O vetor fornecido (sempre par, bit 0 = "0") pelo periférico que interrompeu é um índice nesta tabela, de onde a CPU irá ler o endereço inicial da rotina de serviço pertinente.

Para outras CPU's o esquema é basicamente o mesmo, com pequenas alterações de compatibilização da CPU em questão. Como exemplos de esquemas mais evoluídos, aparecendo em projetos de CPU mais modernos, pode-se citar os esquemas presentes nas CPU's da série 680XX da MOTOROLA [MOTO81] e da série 320XX da NATIONAL/TEXAS [NATI84]. Aqui os apontadores das rotinas de serviço ocupam necessariamente 4 bytes, o que traria uma redução no número máximo de vetores para 64. Isto é solucionado nestes casos fazendo com que os vetores, ao serem adquiridos pela CPU, sofram um deslocamento de 2 bits à esquerda com inserção de 2 bits "0", aumentando a capacidade da CPU para 256 vetores diferentes.

3.3.3 Detalhes das modificações no hardware

A implementação de um novo ciclo de acesso à via SME, para aquisição dos vetores de interrupção, envolveu lógica localizada na placa de CPU decidindo se o vetor será realmente adquirido da via ou do barramento interno da placa, visto que é aí que se conectam os controladores de interrupção/temporizadores. Detalhes desta lógica podem ser vistos na figura 3.5. Outra lógica envolvida na realização do ciclo fica localizada na placa de interface série, já que o número delas pode ir tipicamente a 5, o que pode causar conflitos durante o ciclo de aquisição do vetor de interrupção. Esta lógica, que pode ser vista na figura 3.6, utiliza dois pinos do conector que implementam uma cadeia de prioridades ("daisy chain"). A figura 3.3 mostra a restrição quanto à posição física das placas de interface série com relação à de CPU, a fim de constituírem a cadeia de prioridades, juntamente com "straps" localizados na placa traseira.

Funcionamento da cadeia de prioridades: aos canais de temporização/interrupção (Z80-CTC), localizados na placa de CPU, foi conferida a prioridade mais alta na cadeia. As placas capazes de gerar vetores, quando não estão requisitando interrupção, devem propagar para o sinal /SPR o estado presente em sua entrada /EPR. Quando uma placa ativar o sinal de interrupção vetorizada (BINT7) deve também colocar o sinal /SPR em "0". Quando a placa de CPU iniciar o ciclo de aquisição de vetor, esta placa somente poderá colocar seu vetor no barramento de dados se sua entrada /EPR

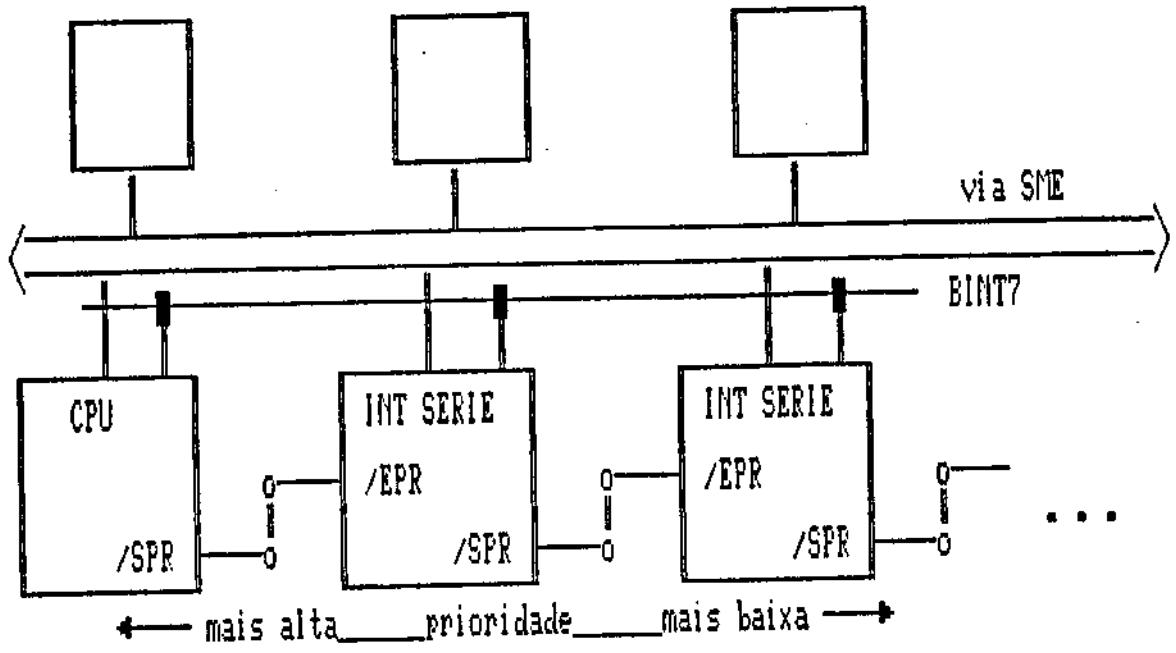


Figura 3.3 : detalhe da cadeia de prioridades de interrupcao

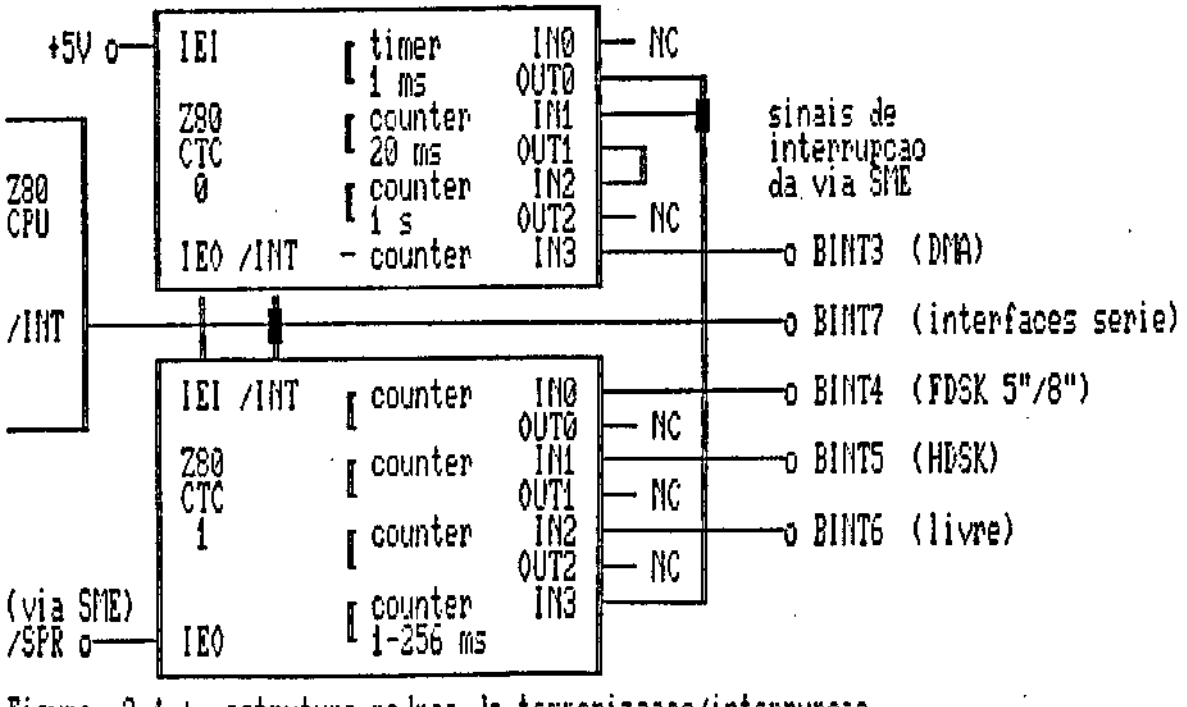


Figura 3.4 : estrutura padrao de temporizacao/interrupcao

estiver em "1".

3.3.4 Configuração de temporizadores/entradas de interrupção

A configuração de temporizadores e entradas de interrupção é definida por "straps", mas as necessidades dos diversos sistemas em implementação permitiram definir uma configuração padrão atendendo todos estes sistemas, a qual pode ser vista na figura 3.4. Cada canal de CTC pode ser programado independentemente como temporizador ou entrada externa de interrupção, além de poder ser individualmente habilitado a gerar interrupções. Normalmente um sistema irá utilizar um subconjunto dos 8 canais disponíveis.

3.3.5 Facilidades da vetorização de interrupções

As facilidades introduzidas pela vetorização de interrupções são bem exemplificadas pelas placas de interface série, onde os "chips" Z80-SIO podem gerar até 4 vetores diferentes por canal, facilitando e agilizando as rotinas de serviço associadas às tarefas de tratamento de entrada/saída. Esses vetores correspondem aos seguintes eventos:

- caracter recebido disponível

- "buffer" de transmissão vazio

- condições especiais de recepção (erros e anomalias)

- mudanças de estado das linhas externas (controle)

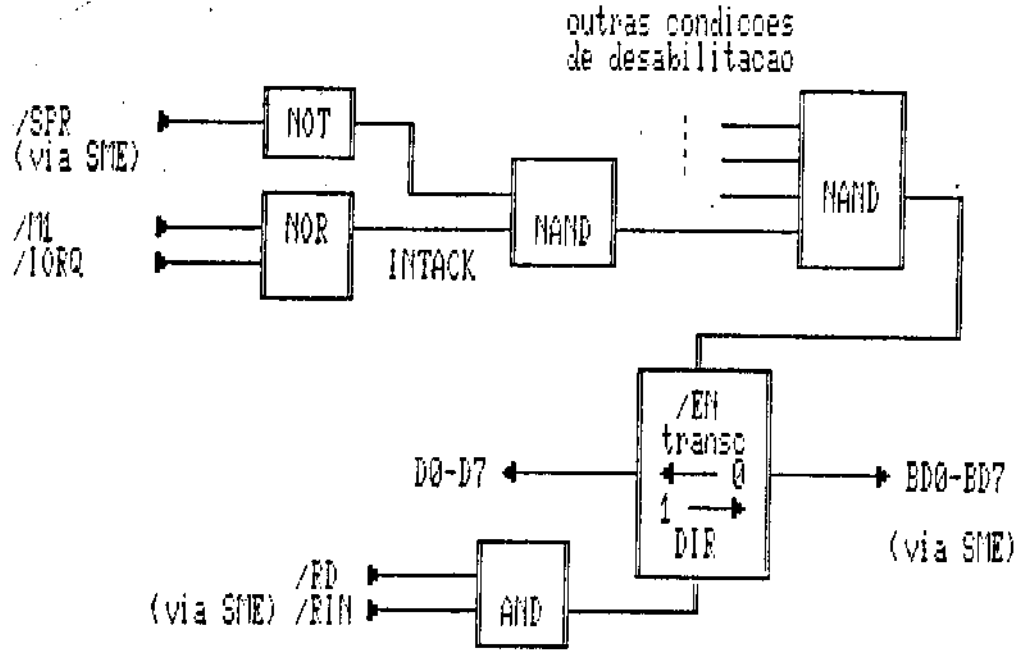


Figura 3.5 : logica de apoio a interrupcoes na placa de CPU

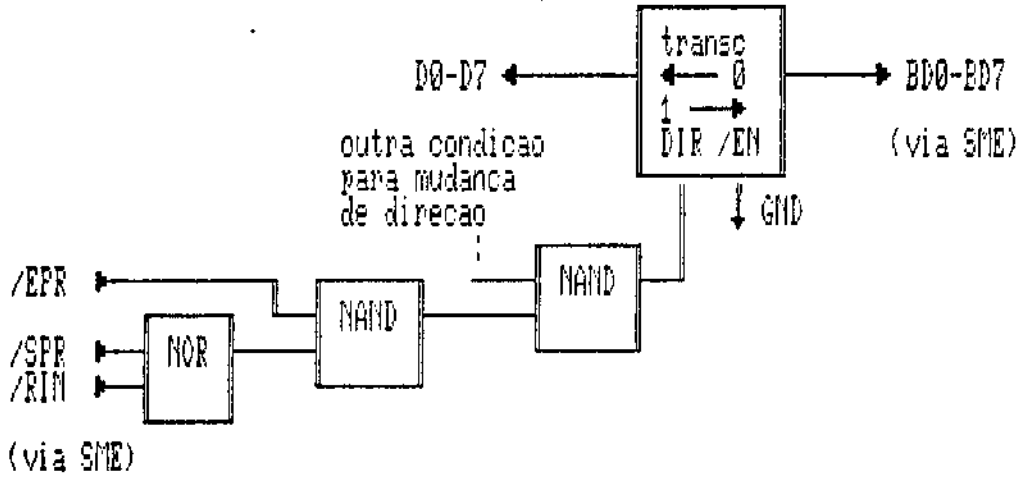


Figura 3.6 : logica de apoio a interrupcoes na placa de interface serie

A otimização das rotinas de recepção está em dois pontos: primeiro, não é necessário "polling" do estado das diversas interfaces que pode ter um sistema; segundo, a rotina de recepção, que é sempre mais demorada em virtude das análises de exceções necessárias, sofre de uma ineficiência crônica porque as exceções, apesar de raras, devem ser sempre inquiridas sobre sua ocorrência. Com vetorizações distintas a rotina de caracter recebido disponível pode ficar restrita ao seu sentido original, e conseqüentemente mais rápida. Qualquer exceção irá gerar um vetor diferente, desviando a execução para uma rotina específica.

CAPÍTULO 4

CONTROLADOR DE DMA

Este capítulo compara métodos de atendimento a eventos assíncronos, destacando vantagens e desvantagens. Compara também os tempos de ocupação da via e justifica a implementação de um grande número de canais de DMA. Apresenta ainda detalhes de hardware.

Um canal de acesso direto à memória não é interessante apenas por uma alta taxa de transferência de dados, mas também por seu baixíssimo tempo de resposta (tempo decorrido entre a requisição e o atendimento) e baixa taxa de utilização de CPU e de barramento.

4.1 Comparação de métodos de atendimento

Para exemplificar tomaremos o caso de uma CPU Z80 a 4 MHz e um controlador de DMA do tipo 8237 à mesma velocidade, examinando três métodos para o atendimento de eventos assíncronos: "polling", interrupção e DMA. Cada evento será representado por duas portas de E/S, uma para status e outra para recepção de dados. Supõe-se também que não haja "wait-states" nos ciclos de acesso [Z80D81].

4.1.1 "Polling"

A CPU fica totalmente dedicada a esta tarefa, continuamente lendo status e verificando se o processo necessita atendimento. Rotinas típicas ficariam assim:

```
                                                        ciclos de
                                                        relógio
PORTA1: IN    (A),STATUS1                                   11
        BIT   BN1,A             ;TESTA EVENTO               8
        JP    Z,PORTA2          ;NÃO, PRÓXIMO              10
        LD    DE,POINTLOC1      ;LOCALIZAÇÃO APONTADOR     20
        LD    A,(DE)            ;APONTADOR EM HL            7
        LD    L,A                                           4
        INC   DE                                            6
        LD    A,(DE)                                        7
        LD    H,A                                           4
        IN    (A),DADO1         ;RECEBE DADO               11
        LD    (HL),A            ;ARMAZENA NO BUFFER         7
        INC   HL                ;INCREMENTA HL              6
        LD    A,H               ;RESTAURA APONTADOR         4
        LD    (DE),A                                        7
        DEC   DE                                            6
        LD    A,L                                           4
        LD    (DE),A                                        7
;
PORTA2: IN    (A),STATUS2
        .
        .
        .
;
PORTAN: IN    (A),STATUSN
        .
        .
        .
        JP    PORTA1            ;INICIO
```

Considerando-se que a CPU esteja apenas executando código referente ao "polling" propriamente dito, vemos que para 1 evento o atendimento demorará no mínimo 19,5 us e no máximo 26,5 us, e para N eventos teremos no mínimo 19,5 us e no máximo ((N x 32,5) + 22) us.

Desvantagens: tempo de resposta ruim, agravado proporcionalmente ao número de processos; para a CPU executar outras tarefas concorrentemente deve ser realizado um "polling" temporizado, definindo a priori o tempo de resposta, sempre pior que no caso com tarefa única.

4.1.2 Interrupção

A CPU pode executar outras tarefas concorrentes, gastando tempo estritamente com o tratamento dos eventos. Supondo-se interrupções vetorizadas com vetores distintos para cada situação (como o chip Z80-SIO), uma rotina de serviço típica é descrita a seguir, relacionando-se os tempos para execução das instruções envolvidas tanto no reconhecimento como na própria rotina de serviço:

```
                                                    ciclos de
                                                    relógio
EI                                                      x
.                                                       .
.                                                       .
.                                                       .
XXX                  ;INSTANTE DA INTERRUPÇÃO           x
(reconhecimento, vetor)                                 7
(salva PC)                                              6
(leitura do apontador)                                  6
(rotina de interrupção ..., SERV)
```

```
                                                        ciclos de
                                                        relógio
SERV:   EXX                 ;SALVA REGISTROS              4
        EX    AF,AF'                                      4
        LD    DE,POINTLOCN  ;LOCALIZAÇÃO APONTADOR        10
        LD    A,(DE)        ;APONTADOR EM HL              7
        LD    L,A                                         4
        INC   DE                                          6
        LD    A,(DE)                                      7
        LD    H,A                                         4
        IN    A,(DADON)     ;RECEBE DADO                  11
        LD    (HL),A        ;ARMAZENA NO BUFFER           7
        INC   HL            ;INCREMENTA HL                6
        LD    A,H           ;RESTAURA APONTADOR           4
        LD    (DE),A                                      7
        DEC   DE                                          6
        LD    A,L                                         4
        LD    (DE),A                                      7
        EX    AF,AF'        ;RESTAURA REGISTROS           4
        EXX                                               4
        EI                  ;HABILITA INTERRUPÇÕES        4
        RETI                ;RETORNO                      14
        (restaura PC)                                     6
                                                        -------
                                              ciclos:     148
                                        total por byte:   37 us
```

O tempo de resposta não depende do número de processos a tratar, mas sim de quantos eventos ocorrem simultaneamente, enfileirando as rotinas de tratamento. O atendimento poderá

levar de 37 us a (N x 37) us para N eventos simultâneos.

4.1.3 DMA

A CPU pode executar outras tarefas com maior disponibilidade, sendo que o atendimento é realizado entre 1,5 us a (((N-1) x 2,2) + 1,5) us, sem a participação da CPU, e desde que haja canais de DMA suficientes para atender as N interfaces causadoras desses eventos.

transferência de 1 byte,
ciclo "flying-by" ---------- 1,0 us

"arbitragem" (tomada e
devolução do barramento ---------- 0,5 us

---

total por byte ---------- 1,5 us

Desvantagem: o gerenciamento dos canais de DMA e dos "buffers" de dados torna-se mais elaborado caso não se conheça de antemão o número de eventos que irão ocorrer para um dado processo. Vantagens: um bom tempo de resposta; possibilidade de a CPU executar outras tarefas concorrentemente; muito "confortável" na transferência de blocos de informação, de tamanho pré-definido.

4.2 Comparação dos tempos de ocupação da via

Como mencionado anteriormente, o número de canais de comunicação série poderia chegar facilmente a 10 em algumas aplicações. Mesmo que as taxas de transferência sejam baixas, o tempo gasto pela CPU somente para tratamento dos canais deve ser multiplicado pelo número de canais, resultando em uma baixa taxa de disponibilidade da CPU. Se outras tarefas devem ser levadas concorrentemente, é vital dispor-se de tempo livre de CPU.

Comparemos a recepção de dados por interrupção e por DMA, para pacotes de tamanho pré-definido, de 10 canais a 19200 baud (ver tabelas de tempo da seção anterior). Considerando-se somente o tempo gasto com a recepção propriamente dita teríamos (caracteres de 8 bits, paridade, 1 stop bit):

periodicidade em
cada canal ---------- 1/19200 bit/s x 11 bits
= 573 us/byte

tempo para a recepção
de 256 bytes a 19200 baud
de 10 canais concorrentes ---------- 256 bytes x 573 us/byte
= 147 ms

tempo de CPU necessário
utilizando-se interrupção ------ 10 x 256 bytes x 37 us/byte
= 95 ms
ou: 64 %

tempo de barramento necessário
(perda de tempo de CPU)
utilizando-se DMA ---------- 10 x 256 bytes x 1,5 us/byte
= 3,8 ms
ou: 2,6 %

4.3 Detalhes de hardware

A figura 4.1 mostra o diagrama de blocos da placa controladora de DMA. Os "chips" LSI controladores de DMA disponíveis implementam de 1 a 4 canais somente, e desejava-se dotar a placa de pelo menos 10 canais, para cobrir um bom leque de aplicações. Com 3 "chips" obtém-se 12 canais (cada canal utilizando dois sinais extra-barramento, DREQ e /DACK), mas apenas 22 pinos do

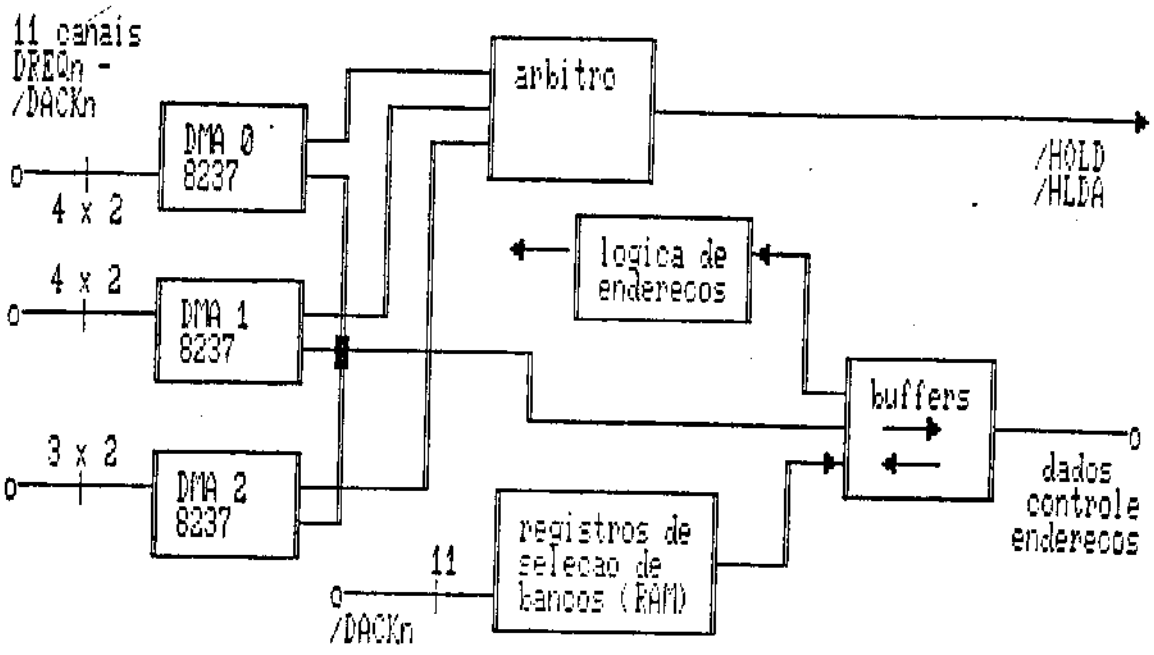


Figura 4.1 : arquitetura da placa controladora de DMA

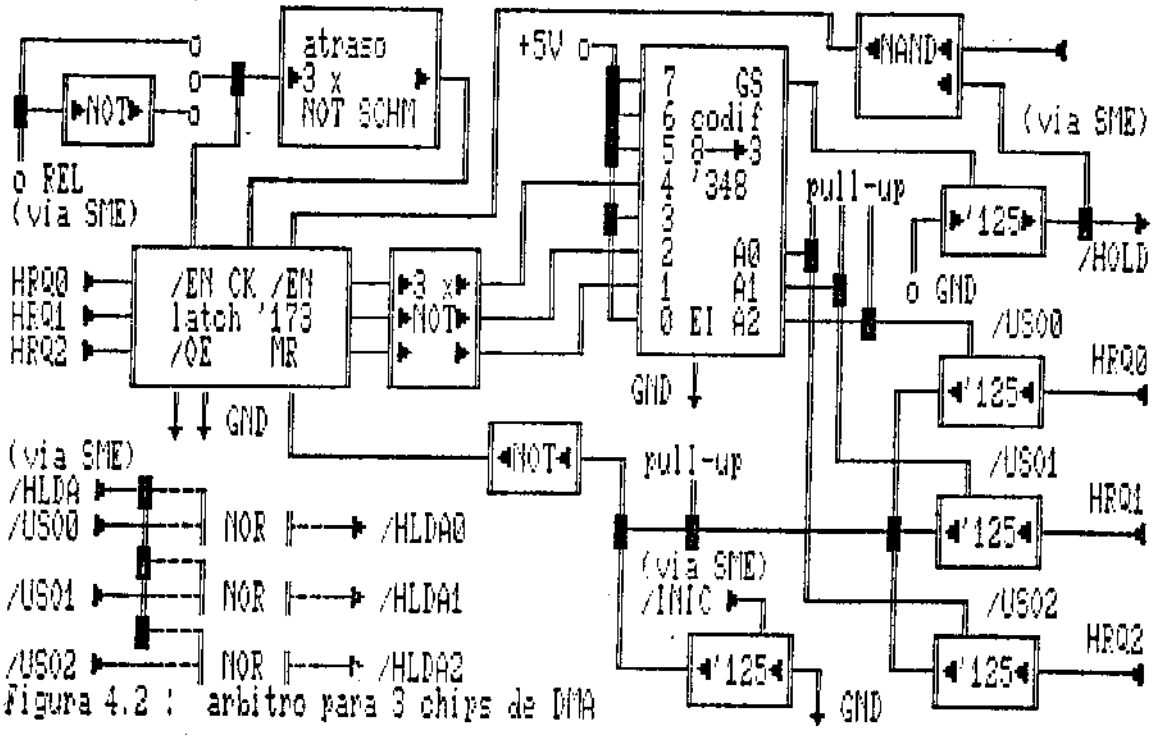


Figura 4.2 : arbitro para 3 chips de DMA

conector estavam disponíveis para uso geral; assim, o número de canais foi limitado a 11. As requisições para tomada de barramento proveniente dos 3 "chips" são gerenciadas por um árbitro construído em lógica discreta. Na figura 4.2 temos o circuito deste árbitro. A possibilidade de inverter o sinal REL permite a inclusão de outra placa de DMA, caso sejam necessários mais canais de DMA. O "latch" '173 bloqueia novas requisições dos "chips" de DMA enquanto a requisição presente não é atendida e completada, isto é, até que /HOLD e /HLDA fiquem inativos. Note-se que o "chip" 0 (HRQ0) é o mais prioritário. O "latch" só captura novos pedidos em uma das bordas do sinal REL, conforme a posição do "strap".

O codificador '348 traduz qualquer pedido HRQn em /HOLD na via SME, ao mesmo tempo que ativa apenas um sinal /USOn, correspondente ao pedido HRQn mais prioritário. Após o atendimento do pedido (desativação do HRQn atendido) o sinal /USOn atendido forçará um "reset" no "latch", retirando o pedido atendido e outros eventualmente capturados mas não atendidos. Um pedido menos prioritário deve portanto permanecer presente (HRQn ativo) até que seja atendido.

O "chip" LSI escolhido (8237, da INTEL) [INTE80] endereça apenas 64 Kbytes, por isso foi necessário dotar a placa de registros de seleção de bancos (4 bits), ampliando o acesso a 1 Mbyte (figura 4.3). Cada canal tem seu registro independente, implementado através de memória super-rápida (2148 de 55 ns). Os sinais /DACK de todos os canais são multiplexados em 4 bits de endereço da memória; o novo

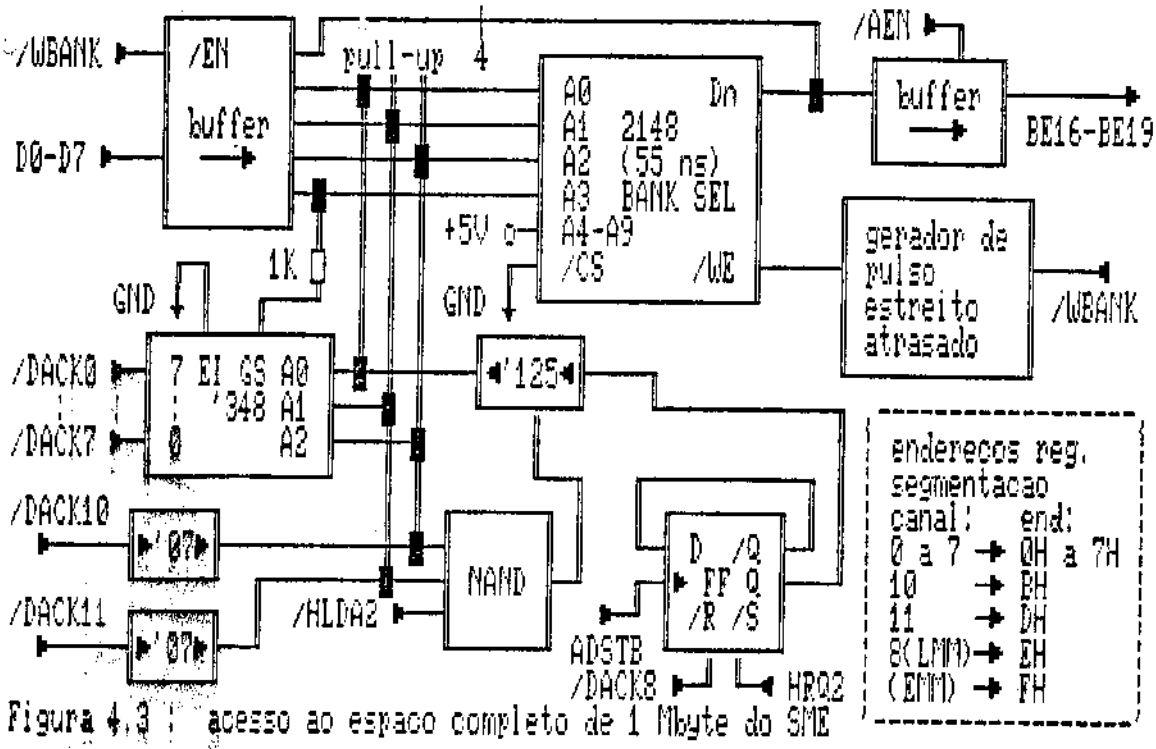


Figura 4.3 : acesso ao espaço completo de 1 Mbyte do SME

ciclo de transferência acrescentado (memória à memória, "flying-through") não utiliza o par de sinais DREQ-/DACK (é comandado por software), mas dispõe de registros de seleção de bancos independentes para a leitura e para a escrita, através de 2 posições da memória 2148.

Cada "chip" de DMA utiliza 16 endereços de E/S, o que daria para a placa: 16 (chip DMA) x 3 chips + 12 registros de seleção de bancos, um total de 60 endereços de E/S, o que causaria na verdade o mapeamento de 64 endereços da via SME, devido a razões práticas de decodificação. Para não restringir demais os endereços de E/S na via, para o caso de CPU de 8 bits (total de 256), construiu-se uma lógica na placa de forma a mapear apenas 16 endereços da via, valendo-se de "buracos" na decodificação interna do "chip" 8237. Pelo "data sheet" do "chip" 8237 pode-se ver que o endereço EH interno ao "chip" está vago, sendo então escolhido para mapear o acesso à memória de segmentação (/WBANK), utilizando-se 4 bits de dados para selecionar o registro desejado (figura 4.3). Um "latch" gera os "chip select" para os três "chips" de DMA, e seu acesso é mapeado no endereço CH interno ao "chip" (acessos a este endereço provocam ações corretivas, e não destrutivas, sem efeitos colaterais). O resultado final é que, em um determinado instante, a CPU pode "ver" o "latch" no endereço CH, os registros de segmentação no endereço EH, e os registros internos do "chip" de DMA (selecionado pelo "latch") nos endereços restantes de 0H a FH, totalizando 16 endereços de E/S do microcomputador SME.

CAPÍTULO 5

RESULTADOS OBTIDOS EM SISTEMAS IMPLEMENTADOS

Neste capítulo são feitas breves descrições dos sistemas implementados utilizando este microcomputador, apresentando a seguir resultados de medidas de desempenho realizadas em sistemas instalados e em funcionamento.

5.1 Sistema USCE

5.1.1 Descrição sumária das tarefas [USCE85]

Este sistema supervisiona uma central telefônica monitorando os estados de seus órgãos (relés), através de conexões especialmente preparadas. Os pontos supervisionados (entradas digitais) têm filtros de 25 ms e sofrem varreduras também de 25 ms. Alguns tipos de órgãos têm apenas um ponto de supervisão, chamado ponto de ocupação. A transição do estado "0" para o estado "1" é chamada "ocupação". O tempo em que o ponto permanece no estado "1" é chamado "tempo de ocupação". Outros tipos de órgãos têm dois pontos de supervisão: o ponto de ocupação e o ponto de eficiência. O primeiro é semelhante aos dos outros órgãos, e o segundo tem definições análogas para "eficiência" e "tempo eficiente". As tarefas de processo, com as respectivas periodicidades, são as seguintes:

- TAR41, 25 ms : - aquisição de dados dos pontos de ocupação dos órgãos de controle rápidos (OR) e dos pontos de eficiência dos órgãos de controle lentos (OL) e dos OR.

- deteção de eventos de ocupação dos OR.

- deteção de eventos de eficiência dos OR e OL.

- TAR42, 250 ms: - aquisição de dados dos pontos de ocupação dos OL.

- deteção de eventos de ocupação dos OL.

- TAR43, 2 s: - acumulação de tempos de ocupação dos OR.

- TAR44, 1 s: - aquisição de dados dos pontos de ocupação dos órgãos juntores (OJ) e dos órgãos seletores (OS).

- deteção de eventos de ocupação dos OJ e OS.

- TAR45, 5 s: - acumulação de tempos de ocupação dos OL.

- aquisição de dados dos pontos de eficiência dos OJ.

- deteção de eventos de ocupação dos OJ.

- TAR46, 20 s: - acumulação de tempos de ocupação dos OJ e OS.

- TAR47, 50 s: - acumulação de tempos de eficiência dos OJ.

- TARJEN, 20 s: - deteção de congestionamento de chamadas dos OJ.

- ERELOG, 1 s: - tarefa de atualização do relógio do sistema e disparo das tarefas temporizadas que rodam em prioridade mais baixa (sem requisitos de tempo real). Ex: relatórios automáticos, análise de HMM (horário de maior movimento).

5.1.2 Outros detalhes

Além das tarefas citadas, o sistema gerencia a interface de comunicação com o sistema UCR (1200 baud) e as interfaces para conexão de um terminal de vídeo/teleimpressora e uma impressora a 4800 baud. O sistema pode também armazenar relatórios/dados em diskette concorrentemente. A configuração atual do sistema USCE pode ser vista na figura 5.1.

As tarefas numeradas de TAR41 a TAR47 têm prioridades associadas mais altas quanto maior a periodicidade. Uma sequência típica de execução de tarefas pode ser vista na

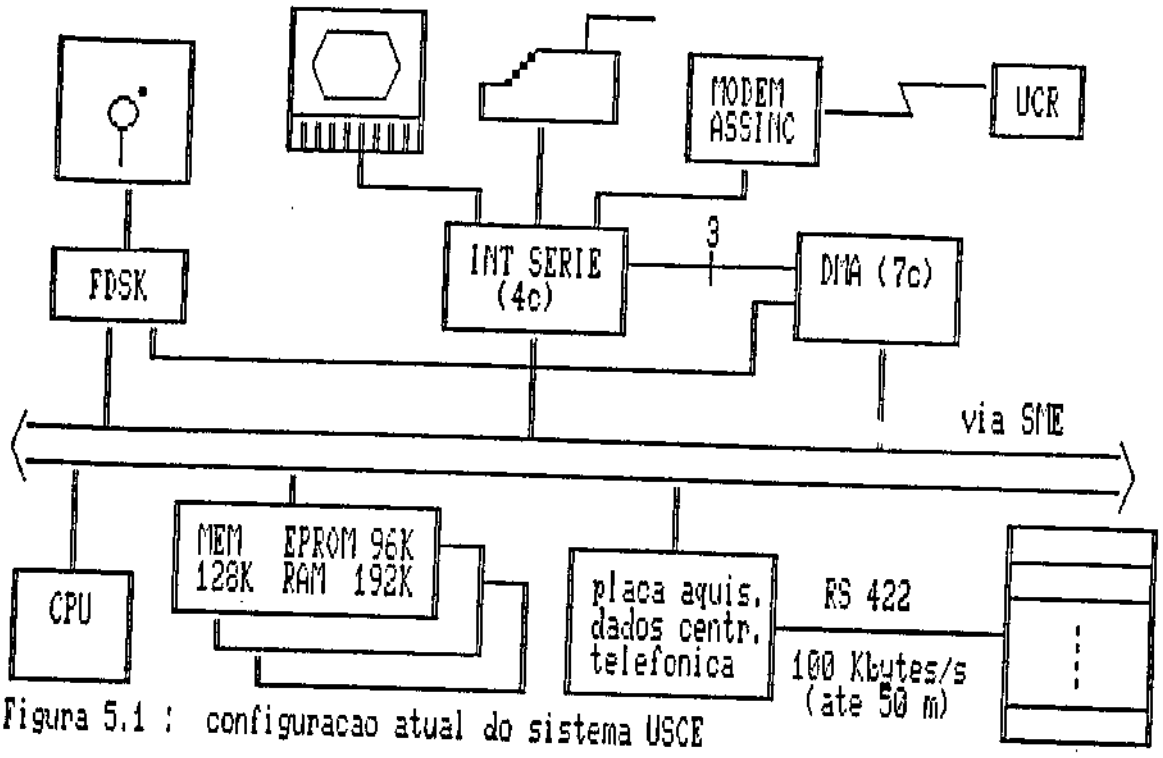
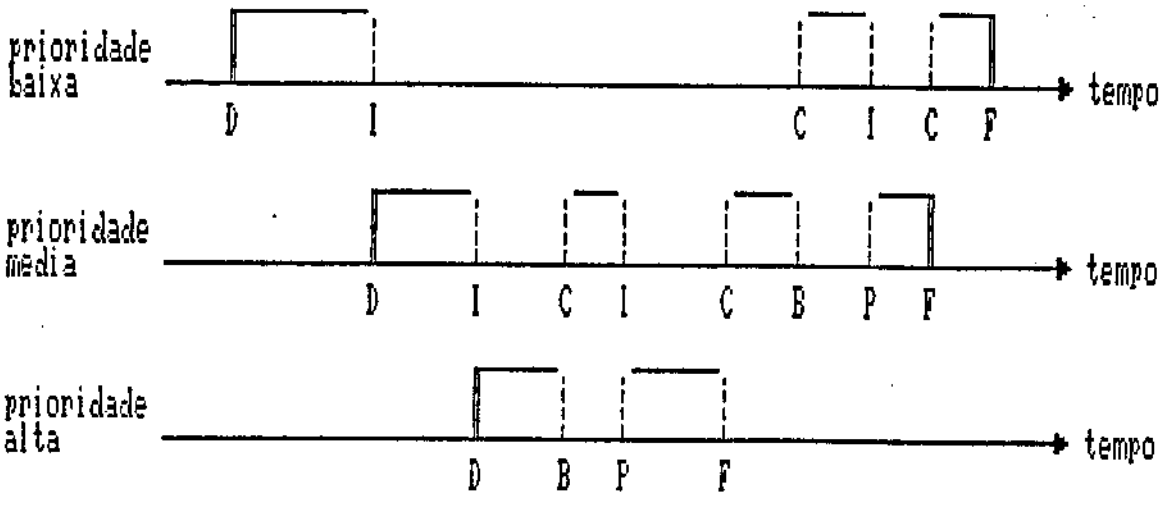


Figura 5.1 : configuracao atual do sistema USCE



D: execucao disparada
I: interrupcao por tarefa mais prioritaria
C: continuacao apos interrupcao
B: bloqueada esperando E/S
P: requisicao E/S pronta
F: final de execucao

Figura 5.2 : execucao de tarefas com prioridades em ambiente multiprogramado

figura 5.2 [SCH074], de conformidade com os requisitos citados na seção 3.1, item "tempo de resposta curto".

O software completo pode ocupar até 288 Kbytes divididos em:

- 96 Kbytes de EPROM para programas.

- até 192 Kbytes de RAM para área de dados.

São utilizados no total 38 contextos da MMU.

5.2 Sistema UCR

5.2.1 Configuração do SME para o sistema UCR

A estrutura básica de configuração do sistema UCR é vista na figura 5.3 [UCRM85]. A equipagem de placas do microcomputador SME (núcleo do sistema) é a seguinte (sistema instalado em Campinas, SP):

- 1 placa CPUZV (CPU Z8C com MMU).

- 1 placa DMAMM (controladora de DMA de 11 canais), sendo utilizados 9 canais.

- 4 placas IS4C (interface série com 4 canais), sendo utilizados 11 canais, com conexões para:

  - 5 USCE's locais.

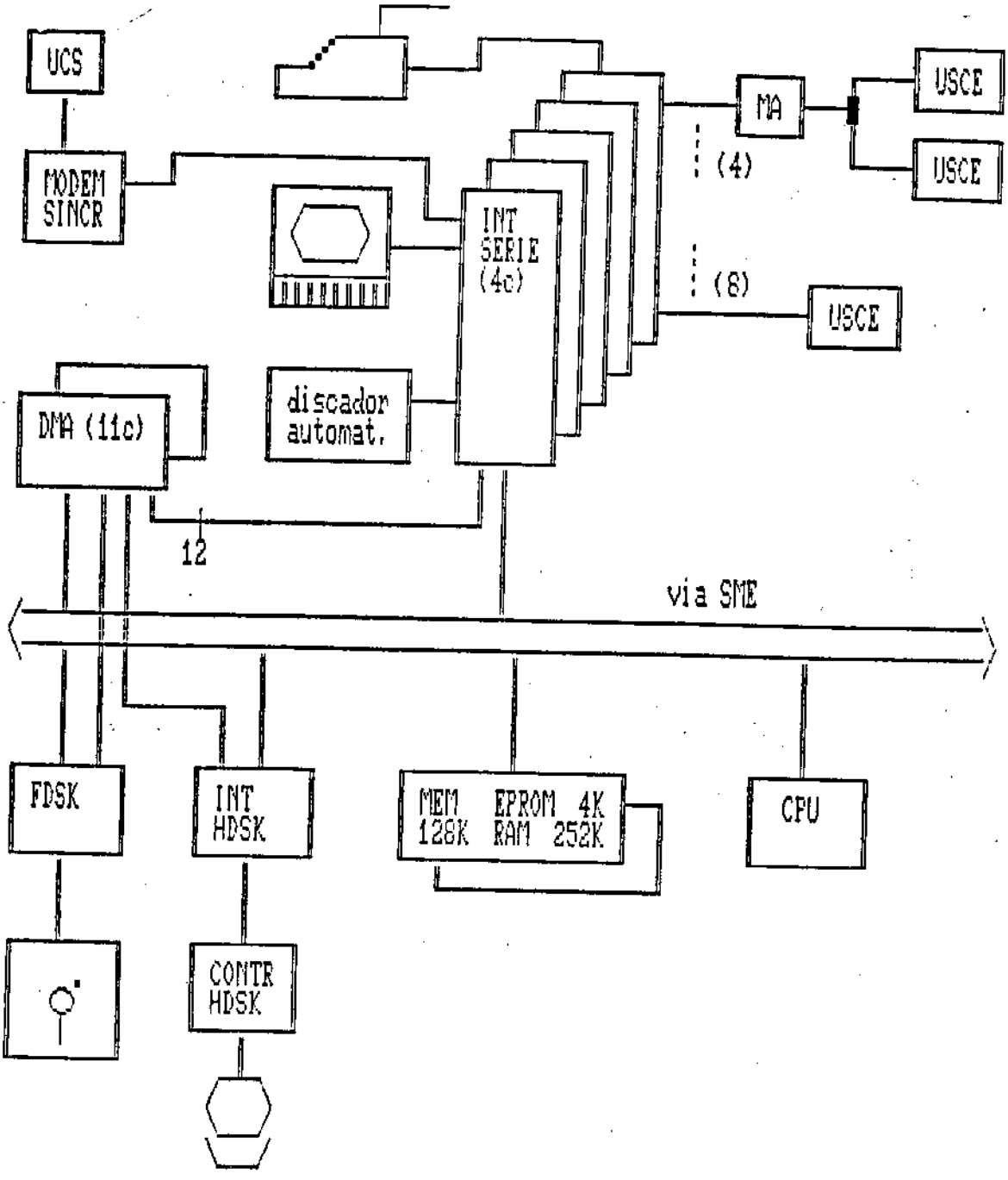


Figura 5.3_1 configuracao basica do sistema UCR

- 2 USCE's remotas conectadas em configuração multiponto através de linha dedicada ("full-duplex") e modem assíncrono a 1200 baud.

- 1 USCE remota conectada através de linha discada ("half-duplex"), discador automático e modem assíncrono a 1200 baud.

- sistema UCS (nível superior) conectado através de linha discada e modem síncrono com respondedor automático a 2400 baud.

- 1 impressora serial.

- 1 terminal de vídeo.

- 1 discador automático.

- 1 placa DSKF (controladora de diskette de 5 1/4").

- 1 placa WCP (interface para controlador de disco rígido Winchester).

- 7 placas RE32K (RAM/EPROM de 32 Kbytes).

5.2.2 Estrutura de software/hardware

O sistema UCR pode ser logicamente dividido em 2 grandes módulos, RAIZ e TAREFAS. A RAIZ está presente em todos os contextos da MMU utilizados, compartilhados por 30 tarefas,

e ocupa os 24 Kbytes superiores do endereço lógico (A000H-FFFFH). Este módulo engloba também o BDOS/BIOS do sistema operacional SMEDOS (compatível com CP/M) em versão 45 Kbytes, o qual roda concorrentemente sob o executivo de tempo real em prioridade mais baixa. O software completo ocupa 208 Kbytes divididos em:

- 4 Kbytes de EPROM ("power-on", "boot").

- 132 Kbytes de RAM pará programas.

- 72 Kbytes de RAM para área de dados.

São utilizados no total 17 contextos da MMU.

5.2.3 Tarefas do sistema

Sob controle do executivo de tempo real rodam 30 tarefas distintas, sendo associados a cada tarefa:

- um único nível de prioridade de execução perante o executivo. O número da tarefa indica o seu nível de prioridade. A tarefa 0 é a de mais alta prioridade.

- 4 filas (FIFO), enfileirando pedidos de execução da tarefa requisitados por outras tarefas em 4 prioridades diferentes (0, mais alta, 1, 2, 3). Os pedidos de mais alta prioridade são atendidos primeiro, mesmo que tenham sido alocados após um pedido de menor prioridade. Quando o executivo atende uma tarefa (item anterior), ele decide

qual tratamento será dado a esta tarefa (requisitada por uma outra tarefa) avaliando as FIFO's.

Na figura 5.4 estão esquematizadas as chamadas inter-relacionadas das tarefas do sistema.

As ligações não pontilhadas identificam o esquema de chamadas entre tarefas. Os números sobre cada ligação XX/YY representam respectivamente o nível de prioridade da chamada e o número máximo de chamadas, nesta prioridade, aguardando atendimento. XX (barrado) indica chamada com espera, XX (não barrado) indica chamada sem espera.

As ligações pontilhadas indicam chamadas sem passagem de parâmetros (STASK) ou solicitações para despertar determinada tarefa (WAKE). Os triângulos pretos indicam a ocorrência de interrupções que diretamente disparam ou despertam determinadas tarefas. As tarefas de execução cíclica, disparadas pela rotina MYSHED localizada na RAIZ do sistema, são identificadas pela incidência de uma aresta TEMP/XX, onde XX indica a periodicidade da tarefa.

5.3 Resultados de medidas de desempenho

Foram realizadas medidas da porcentagem de tempo em que a CPU permaneceu no endereço de "idle" do executivo dos sistemas UCR e USCE, controlando o barramento (isto é, eventuais transferências de DMA não são contadas como tempo ocioso). O resultado obtido é realmente o tempo ocioso

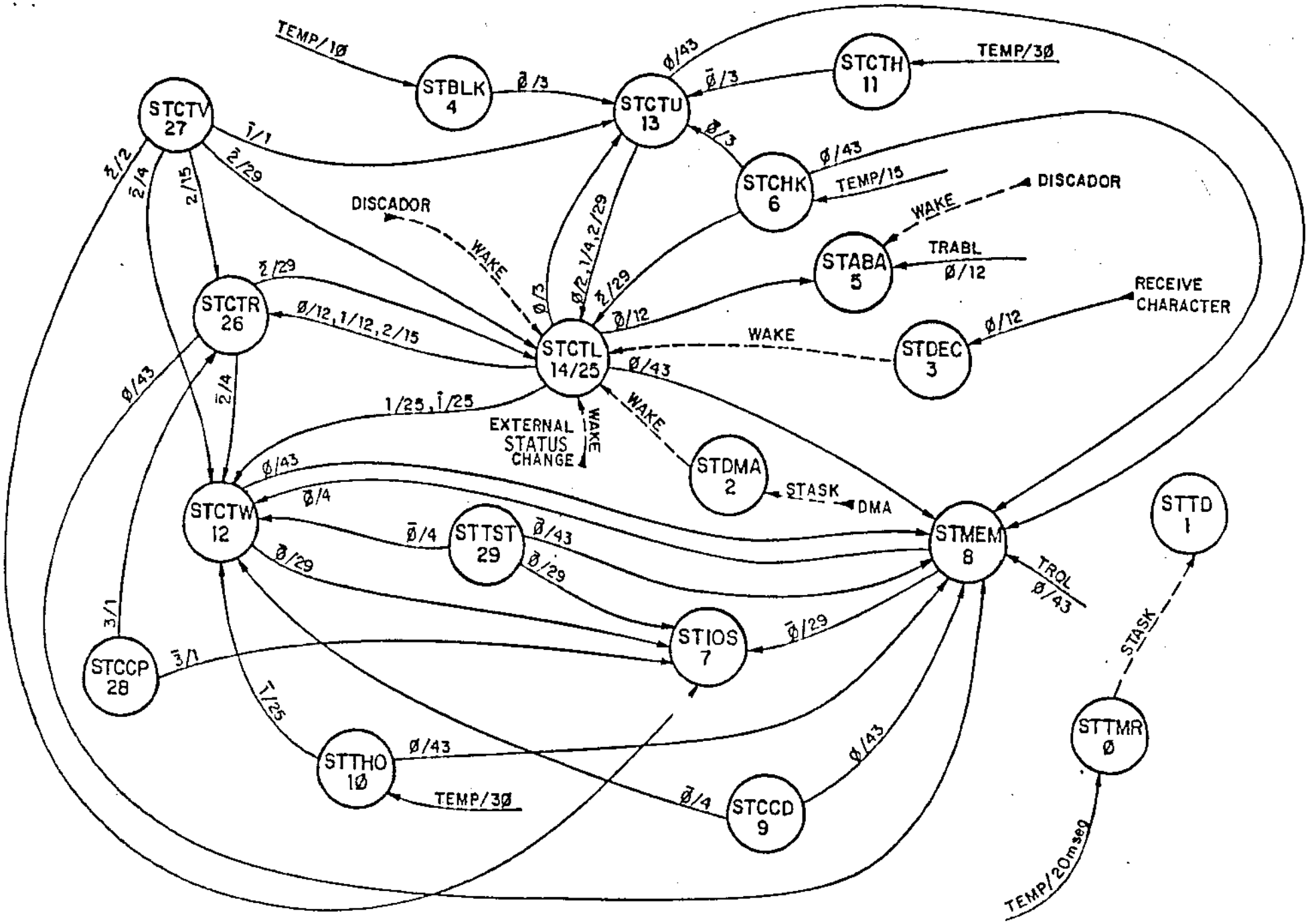


Figura 5.4 : chamadas de tarefas do sistema UCR

líquido, com grande precisão. Um hardware simples de "trigger" configurável permitiu a utilização de um freqüencímetro ao invés de um analisador lógico.

Este hardware é constituído de um conjunto de comparadores de magnitude de 8 bits (74LS682) aos quais estão conectadas as linhas de endereço e de controle da via SME. Através de chaves cada sinal pode ser selecionado a entrar ou não na lógica de comparação, assim como com qual estado ("0" ou "1") será comparado. Quando as saídas dos comparadores estiverem ativas uma lógica habilitará a passagem do sinal REL (4,096 MHz) à entrada do freqüencímetro, cuja leitura poderá ser normalizada e eventualmente corrigida (veja parágrafo seguinte). O controle individual da cada sinal de endereço da via SME permite escolher uma faixa de endereços sobre a qual será feita a análise de desempenho. O tamanho desta faixa pode ser alterado em função de potências inteiras de 2, como pode ser deduzido da lógica acima.

Nas medições realizadas foi selecionado o endereço físico de "idle" dos executivos, o sinal /LEM ativo e o sinal /HLDA inativo, para garantir um acesso de CPU. Foram realizadas eventuais correções relacionadas às janelas de aquisição e às fronteiras reais de endereço das instruções envolvidas.

Foi utilizado um intervalo de amostragem da ordem de 1 seg, com intervalo entre amostragens de 0,2 seg, sendo anotadas em média 1 de cada 4,5 medidas. Como o instrumento de amostragem é assíncrono ao sistema em medição, as

amostragens são uma variável aleatória. Garante-se então, com um número razoável de amostragens, uma boa precisão para o valor médio de carga.

Sistema USCE #1 -- cerca de 14000 pontos de características mais lentas, segunda-feira (dia da semana de maior trânsito na maioria das centrais telefônicas), horário de 09:51 a 09:56:

carga média : 44,8 %

Sistema USCE #2 -- 11800 pontos em configuração média (em termos de velocidade), segunda-feira, horário de 10:04 a 10:21 (horário mais pesado da semana):

carga média : 74,6 %

piores momentos
(universo de 30 a 60 seg) : 80 %

carga média
(horário de 11:51 a 11:54,
"baixa" de almoço) : 69 %

Observa-se que nos momentos em que são disparados relatórios automáticos de baixa prioridade (sem requisitos de tempo real) o sistema trabalha com carga de 100 % , aproveitando as "sobras" das tarefas de tempo real.

O sistema USCE #3 supervisiona quase 16.000 pontos. Conhecendo-se os dados do sistema USCE #2, estima-se que a carga na USCE #3 nos momentos mais críticos deva estar ao redor de 85 % (é conhecida a taxa aproximada de aumento de carga em função do número de pontos)

Sistema UCR #1 -- 8 USCE's conectadas (não há muita dependência com o trânsito nas centrais telefônicas):

carga média : 56 %

oscilações entre : 44 %

e : 69 %

Observa-se que quando o sistema emite relatórios em impressora, ou quando o operador entra em modo SMEDOS (CP/M) concorrente, a carga aparente sobe a 100 %, em virtude de seu modo de funcionamento, já que este sistema operacional roda sob prioridade baixa e é contemplado com todo o tempo ocioso do executivo de tempo real.

Outras medições foram realizadas no sistema USCE pelo grupo de software utilizando-se um traçador de software implementado através de uma tarefa contemplada com a prioridade mais alta. Foram obtidas tabelas de tempos em milissegundos dos intervalos de tempo desde a requisição de uma tarefa até o final de execução, e também desde o início de execução até o final de execução. Verificou-se que os requisitos de tempo real de todas as tarefas são plenamente atendidos, sendo que as tarefas sempre foram executadas

completamente antes de decorrida a metade do tempo para a próxima requisição. A única exceção corresponde à tarefa 41, de 25 ms (ver seção 5.1.1), que corresponde a uma taxa de ocupação média de 40 %, podendo chegar a 65 %. A figura 5.2 mostra o esquema básico sob o qual as tarefas são executadas.

CAPÍTULO 6

OUTRAS APLICAÇÕES E SEUS RESULTADOS

Este capítulo apresenta outras aplicações para este microcomputador, já implementadas ou em desenvolvimento, destacando sua versatilidade e modularidade. Discute também os resultados parciais obtidos.

Como mencionado anteriormente este microcomputador teve como meta em seu desenvolvimento a flexibilidade necessária para se adaptar às diversas aplicações previstas. Nesta seção serão apresentados outros sistemas já implementados ou em implementação que utilizam o microcomputador SME, sendo um sistema de processamento multiusuário, um sistema servidor de arquivos e um sistema de telessupervisão.

6.1 Sistema multiusuário

Este sistema utiliza o sistema operacional MP/M, compatível com CP/M, cuja configuração dispõe de 3 terminais de vídeo, uma impressora, um disco rígido de 5 Mbytes e duas unidades de disco flexível de 8" utilizadas como via de acesso ao sistema. A equipagem completa do sistema pode ser vista na figura 6.1.

A concentração de periféricos em uma única máquina pode proporcionar aos usuários melhores recursos, ao mesmo tempo que diminui o custo global por usuário. Apesar do hardware admitir apenas três usuários simultaneamente, o sistema

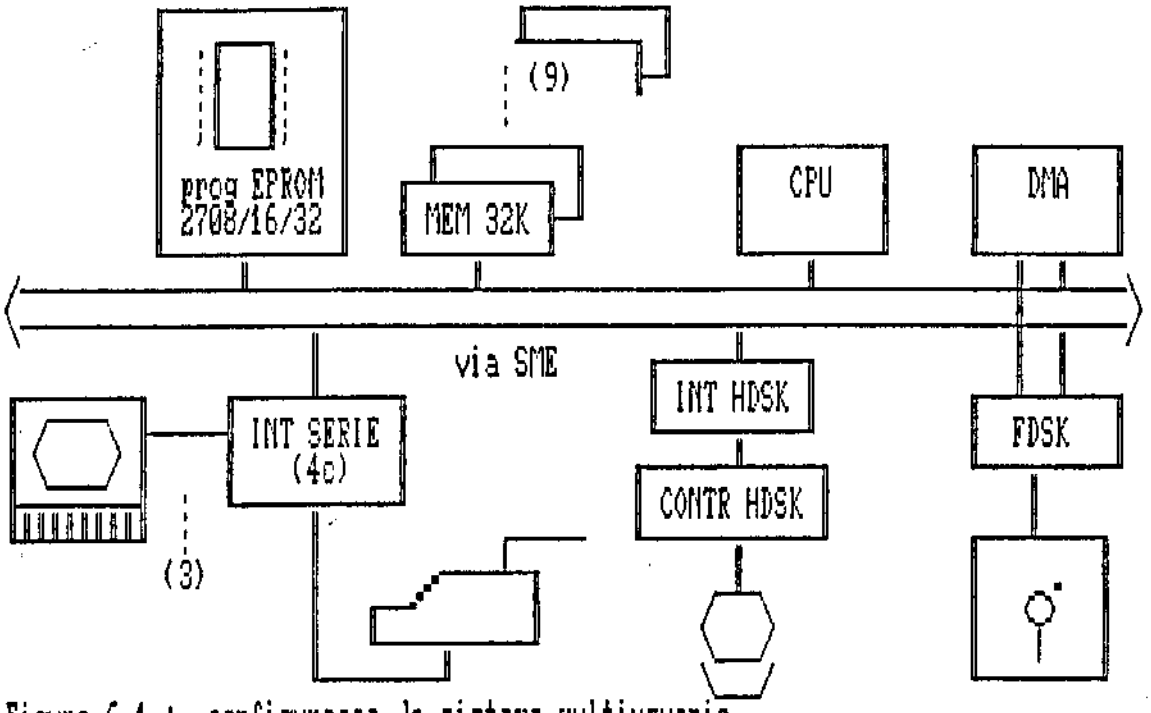


Figura 6.1 : configuracao do sistema multiusuario

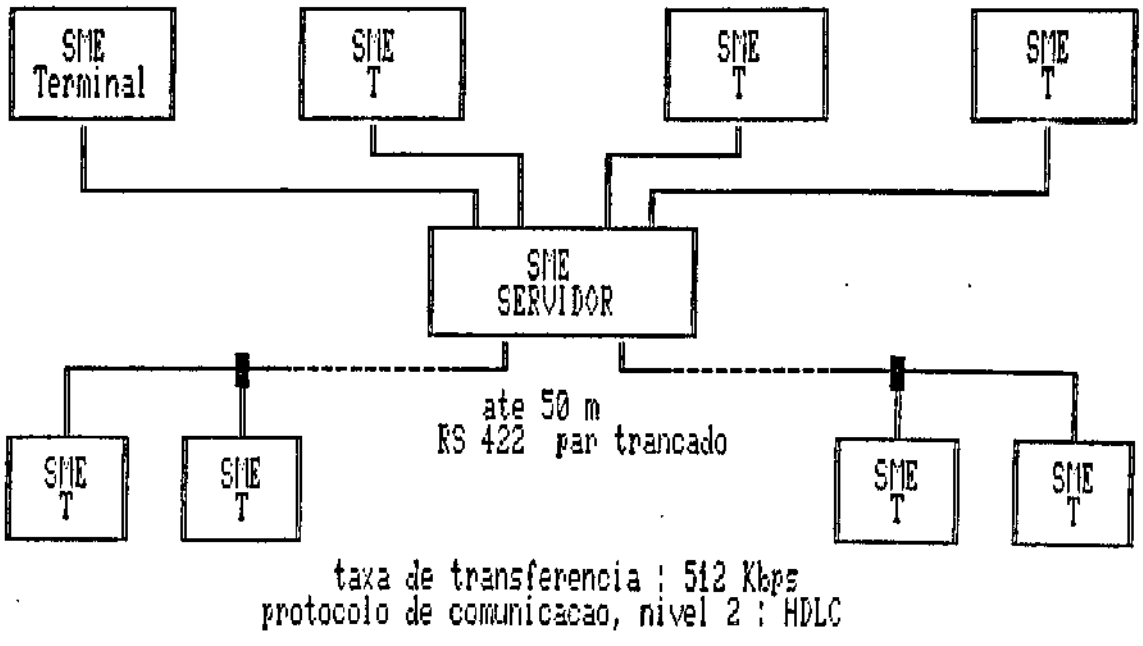


Figura 6.2 : arquitetura da rede servidor - terminais

operacional dispõe de áreas para 15 usuários, podendo atender um bom grupo de usuários supondo-se que cada um gaste apenas parte de seu tempo utilizando um terminal.

6.1.1 Resultados obtidos

A experiência adquirida com a utilização de três destas máquinas permitiu levantar suas deficiências e localizar seus pontos de estrangulamento, assim como levantar conclusões importantes sobre suas possibilidades futuras, relatados a seguir.

- o sistema não consegue alimentar simultaneamente os três terminais (operando a 19200 baud) às suas velocidades máximas; a velocidade efetiva seria equivalente a aproximadamente 6000 baud.

- quando o sistema realiza operações em disco a alimentação de caracteres aos vídeos sofre interrupções visíveis, agravadas em muito quando essas operações envolvem disco flexível, bem mais lento que o disco rígido. Programas de alta utilização de disco como o "PIP" tornam desagradável a utilização da máquina pelos outros usuários.

- a memória de trabalho para cada usuário limitada em 48 Kbytes restringe a utilização de certos programas.

- a baixa velocidade de alimentação dos vídeos e de resposta ao teclado reside basicamente no fato de as chamadas ao sistema operacional para operações de transferência com

vídeos/impressora não serem completamente desvinculadas daquelas para operações em disco, além da ineficiência intrínseca de seu código.

- o sistema é completamente vulnerável a um usuário que esteja depurando programas (depuradores tipo "DDT").

- a maior parte das deficiências pode ser sanada com uma melhor utilização dos recursos da arquitetura, porém isso envolveria modificações mais profundas no sistema operacional, dificultadas pela não disponibilidade do código fonte.

- conclui-se que as vantagens da arquitetura só se tornam reais se o software básico puder explorá-las a contento, como é o caso dos sistemas UCR e USCE vistos anteriormente. A menos que se desenvolva um novo sistema operacional com este intento, o gargalo do sistema multiusuário continuará a residir nas rotinas básicas do sistema operacional.

6.2 Sistema servidor de arquivos

Uma nova arquitetura global foi idealizada baseada principalmente na distribuição do processamento. Trata-se de uma rede em estrela cujo nó central é uma máquina dispondo de um conjunto completo de periféricos e trabalhando como um servidor de arquivos. Os laços de comunicação com os microcomputadores remotos ou terminais

são a priori ponto-a-ponto, mas podendo trabalhar em configuração multi-ponto. A comunicação é síncrona de alta velocidade (utilizando-se RS 422/485),para distâncias de até 50 m. A arquitetura da rede é vista na figura 6.2. A configuração típica terá 8 microcomputadores remotos.

6.2.1 Microcomputador terminal

O microcomputador remoto ou terminal é baseado em um barramento SME de 5 "slots", podendo ser derivado de uma placa traseira padrão de 20 "slots" seccionada em 4 sub-barramentos. A arquitetura interna pode ser vista na figura 6.3. A placa de interface série abriga a conexão síncrona com o servidor, a conexão assíncrona com o terminal de vídeo e uma terceira conexão assíncrona que pode ser utilizada para uma impressora local (setores administrativos) ou um emulador para desenvolvimento de software/hardware. Duas placas de memória de nova geração proverão a máquina com 256 Kbytes, sendo 64 Kbytes destinados ao sistema operacional monousuário e à memória de trabalho, e os 192 Kbytes restantes implementando um disco virtual em memória.

6.2.2 Servidor de arquivos

O servidor de arquivos propriamente dito pode ser visualizado na figura 6.4. A configuração típica exigirá 4 placas de interface série de modo a fornecer 8 canais síncronos com utilização de DMA. Os 4 canais assíncronos restantes abrigarão 2 impressoras, um eventual terminal de

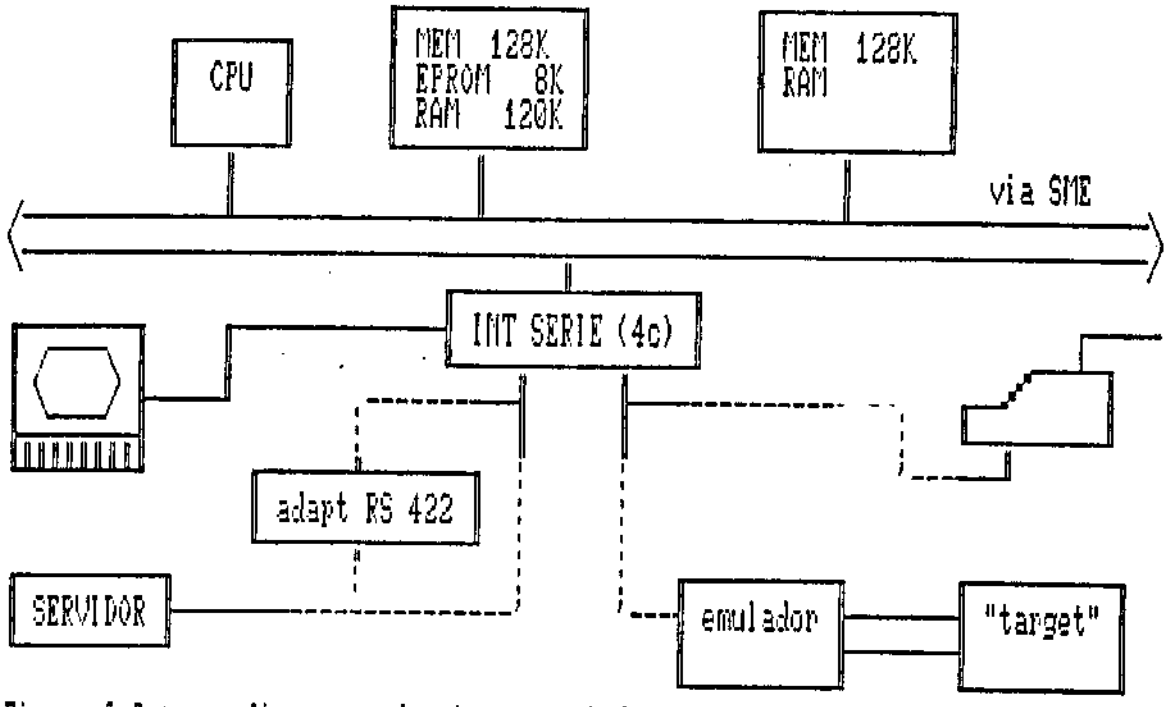


Figura 6.3 : configuracao do microcomputador terminal

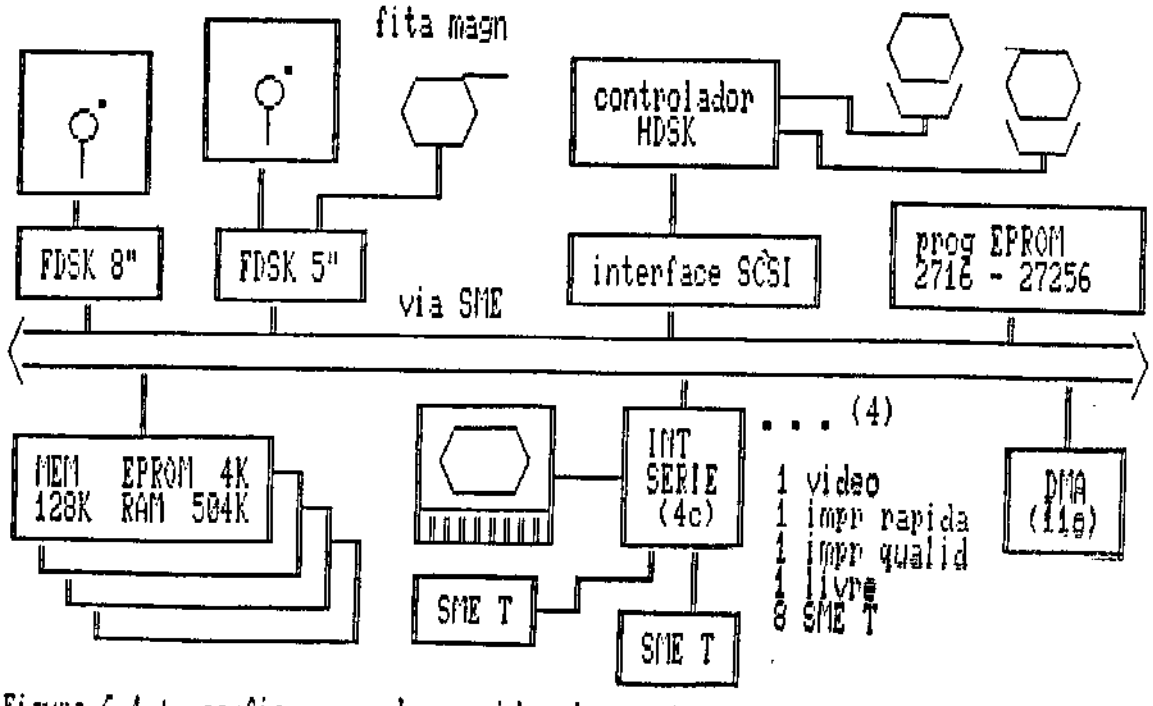


Figura 6.4 : configuracao do servidor de arquivos

vídeo para manutenção e uma conexão suplementar (outros 4 canais não poderão ser utilizados devido à configuração de "straps" a ser adotada). A máquina disporá de controladores de disco flexível de 8" e de 5 1/4 " (este último controlando também uma unidade de fita magnética para "back-up"), de interface para controlador de disco rígido (aceitando unidades de 10 a 60 Mbytes) e de um programador de EPROM (2716/32/64/128/256). As 8 interfaces síncronas serão atendidas por DMA, assim como os controladores de disco flexível e rígido.

6.2.3 Considerações sobre a implementação

Testes preliminares revelaram que a placa de CPU com Z80 a 4,096 MHz, juntamente com as placas de interface série e memória, é capaz de transmitir ou receber dados a uma taxa de até 512 Kbps (1 byte a cada 15,6 us), desde que o tamanho máximo dos pacotes seja igual ou inferior a 256 bytes. Isto permite dispensar a placa controladora de DMA no microcomputador terminal. No servidor de arquivos o número máximo de canais de DMA ativos em um dado instante é 6, limite este ditado pela capacidade do barramento e pela lógica de funcionamento dos controladores de DMA. As facilidades fornecidas pelo "chip" Z80-SIO para a implementação de um protocolo SDLC/HDLC irão garantir transferências livres de erros.

O conhecimento completo do sistema operacional monousuário (compatível com CP/M) permite que se utilize suas seções que interessam de uma maneira mais eficiente no servidor, seções

estas que serão utilizadas somente para operações em memória de massa. A manipulação das conexões síncronas e do sistema como um todo estará a cargo de um executivo otimizado para velocidade. Presentemente já está disponível uma versão do sistema operacional (SMEDOS) que aceita múltiplas operações em disco concorrentemente. Desta forma suas limitações estarão localizadas apenas nos algoritmos de alocação e demais processamentos, além de nos próprios tempos de acesso dos discos.

6.3 Sistema de telessupervisão

Este sistema, denominado SALCOM, está sendo desenvolvido baseado no microcomputador SME. Trata-se de um sistema de telessupervisão e contrôle distribuídos, cuja implementação inicial irá supervisionar uma rede estadual de comunicações baseada em microondas.

As funções são distribuídas em três níveis hierárquicos, sendo que a rede de interconexão dos processadores constitui uma arquitetura mixta anel-radial (figura 6.5), para maior integridade e manutenção do sistema em caso de falha de algum de seus componentes. A degradação na disponibilidade do sistema é gradativa, garantindo seu funcionamento até que as rotas alternativas sejam interrompidas.

A figura 6.6 mostra a configuração de um nó CP ou CR (controlador principal ou regional, respectivamente). A rede de comunicações é gerenciada pelos sistemas CC

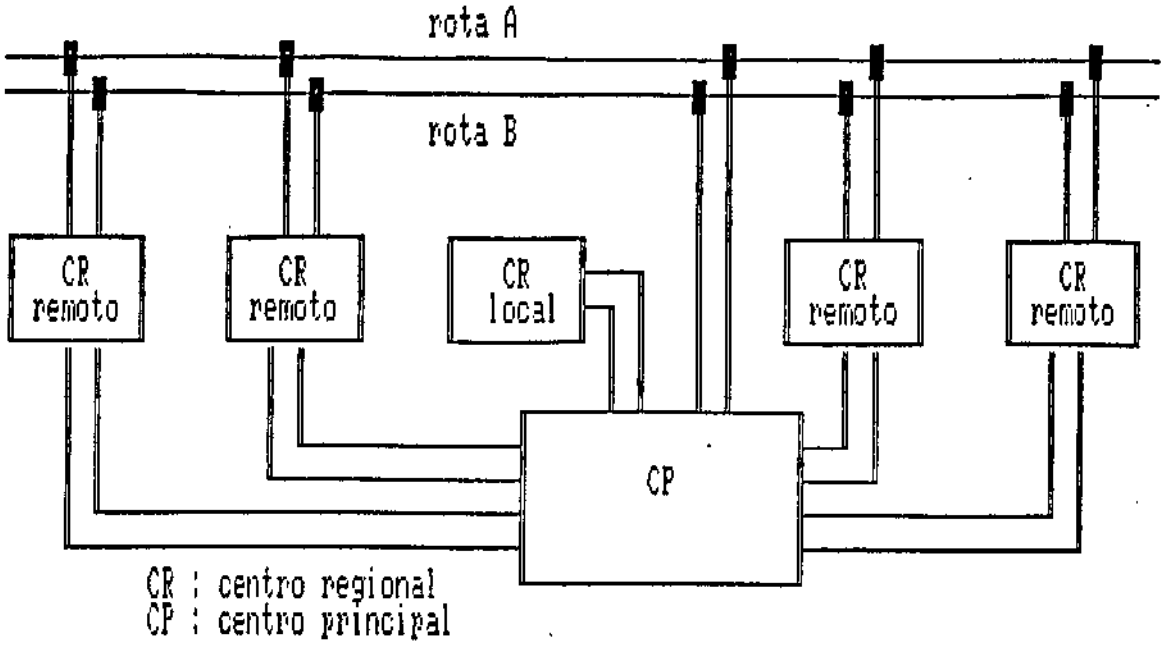


Figura 6.5 : arquitetura de comunicacao do sistema de telessupervisao

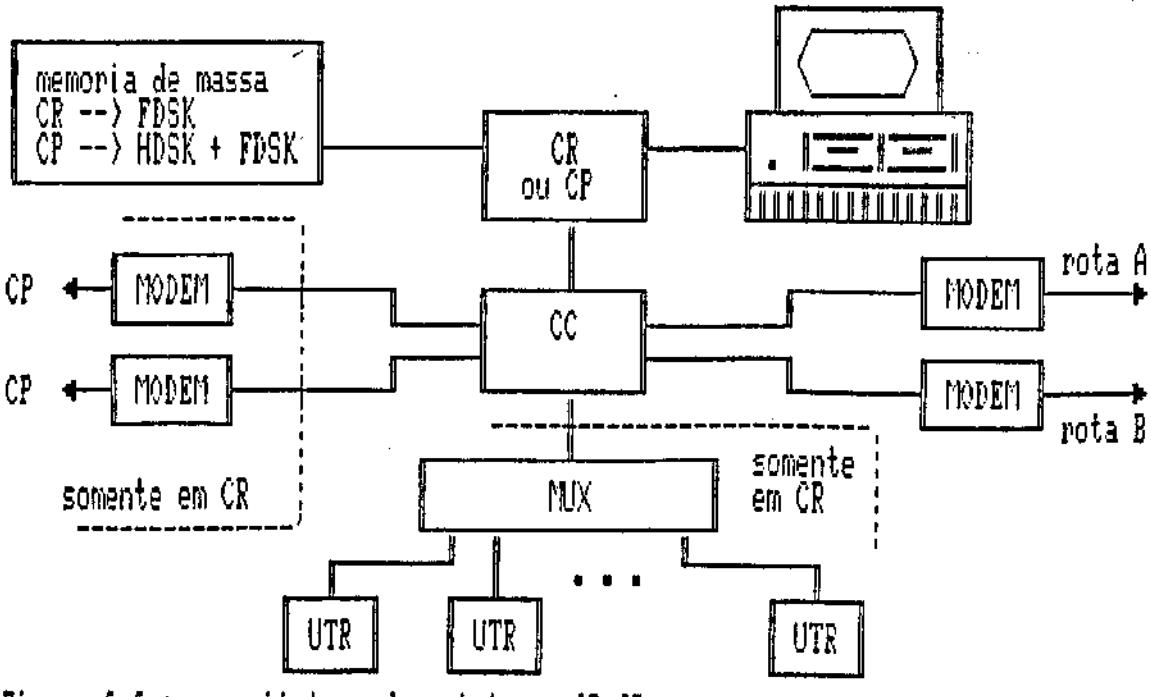


Figura 6.6 : arquitetura dos sistemas CP/CR

(controlador de comunicações) em cada nó. A inteligência de cada nó está nos sistemas CP ou CR e nos microcomputadores de interface homem-máquina. Os sistemas CR, CP e CC são implementados através de diferentes configurações do microcomputador SME, enquanto microcomputadores compatíveis com a linha IBM-PC realizam a interface homem-máquina, valendo-se de seus vídeos gráficos coloridos para uma melhor interação.

A tolerância global do sistema a falhas é garantida por vários pontos: configuração mixta anel-estrela; todos os laços de comunicação são duplicados; todos os sistemas CC são redundantes, garantidos por circuitos especiais de chaveamento de sinais controlados por "watch-dog"; o sistema CP pode ser substituído em suas funções pelo sistema CR local; qualquer sistema CR pode ser substituído em suas funções pelo sistema CP, graças aos laços da rede estrela.

Os sistemas CR concentram sistemas remotos (UTR) dedicados à monitoração e atuação sobre o processo. Um sub-sistema (MUX) permite a conexão de um grande número de UTR's a poucos canais série disponíveis nos sistemas CC, que repassam as informações aos CR's associados. As vias de comunicação entre um CC e uma UTR podem ser uma fônica a 75 baud ou um modem a 150 baud. O sistema sendo implementado terá 5 sistemas CR concentrando no total 373 sistemas UTR.

Os sistemas UTR são constituídos de um microcomputador de placa única (MPU) controlando placas de aquisição de dados e

atuação (do sistema TAJUS) que normalmente são controladas por um microcomputador SME através de sua placa de expansão de E/S. Os tipos de interface básicos são entradas e saídas digitais e entradas analógicas. Os conversores A/D de 12 bits tratarão sinais dos seguintes tipos: DC < 1V; DC > 1V; AC < 250V; DC diferencial < 5V; sinal fornecido por sensor de temperatura.

CAPÍTULO 7

CONCLUSÕES

As modificações realizadas neste microcomputador modular visaram torná-lo capaz do processamento de tarefas concorrentes, sob requisitos de tempo real. Os objetivos procurados foram atingidos através de: gerenciamento de memória para ampliação da capacidade de endereçamento da CPU de 64 Kbytes para 1 Mbyte; implementação de um número adequado de entradas de interrupção, de forma a atender os dispositivos de E/S utilizados em cada aplicação; utilização de um controlador de DMA com um grande número de canais, a fim de poupar tempo de processamento à CPU em tarefas de comunicação.

As principais características de hardware do sistema obtido são: unidade de gerenciamento de memória utilizando um esquema de paginação com páginas de 4 Kbytes, com manutenção de 64 contextos de paginação residentes na unidade; esquema de interrupções vetorizadas possibilitando a identificação automática de interrupções de até 128 eventos diferentes; controlador de DMA dispondo de 11 canais independentes com acesso ao espaço total de 1 Mbyte do microcomputador.

As principais qualidades do sistema inicial (modularidade e expansibilidade) foram preservadas. A maior capacidade de processamento adquirida dotou o sistema de grande versatilidade, como se pode ver pelos tipos de aplicações apresentados nos capítulos 5 e 6 (sistemas UCR e USCE,

sistema multiusuário, sistema servidor de arquivos e sistema de telessupervisão).

O desempenho do microcomputador foi avaliado nos dois sistemas que motivaram este aperfeiçoamento, comprovando-se o atendimento das funções e requisitos de tempo real destas aplicações. Uma outra aplicação (sistema multiusuário) demonstrou que os recursos da nova arquitetura só se efetivam na medida em que o sistema operacional os explora convenientemente.

Durante as atualizações de software destes sistemas implementados tem-se notado dificuldades práticas na modificação e criação de contextos (projeto de software), devido à sua grande quantidade nestas aplicações. Isto é conseqüência direta do reduzido espaço linear de endereçamento, intrínseco à CPU Z80. Estas atualizações de software também são trabalhosas devido a este ser escrito em linguagem "assembly", sem possibilidades de ser transposto para uma linguagem de alto nível ("C", por exemplo) e tornar fácil o seu suporte (não é o caso do sistema UCR, onde 90 % do software é escrito em "C" e somente as partes críticas de velocidade em "assembly"). Esta certeza fica patente observando-se as medidas de desempenho fornecidas, visto que bons compiladores irão causar uma perda de capacidade de processamento da ordem de 3 a 5 vezes.

O próximo passo na evolução deste microcomputador é a introdução de uma CPU de 16 bits, dada a disponibilidade atual de software básico e ferramentas para desenvolvimento.

A escolha de uma CPU com arquitetura interna adequada é fundamental para se obter o ganho de processamento desejado, destacando-se a disponibilidade de um amplo espaço linear de endereçamento para evitar as dificuldades na manipulação de muitos contextos, encontradas na CPU de 8 bits. O ganho de processamento fornecido por tal CPU de 16 bits permitirá o projeto de sistemas com software escrito totalmente em alto nível, e apenas eventualmente necessitar de otimizações em partes críticas. A compatibilidade com a via SME permitirá a utilização de todos os recursos funcionais de hardware existentes.

BIBLIOGRAFIA

BRON82 - BRON, C., DIJKSTRA, E. J. e SWIERSTRA, S. D. A memory-management unit for the optimal exploitation of a small address space. Information Processing Letters 15, 1 (19/Ago/1982), 20-22.

DILL82 - DILLON, B. Extended addressing for the Z80. Wireless World 88, 1559 (Ago/1982), 60.

GUIM79 - GUIMARÃES, C. C. Princípios de Sistemas Operacionais. I.M.E. da USP, São Paulo, SP, 1979.

INTE80 - 1981 COMPONENT DATA CATALOG. Intel Corporation, Santa Clara, CA (1980), 6/91-6/105.

JENS74 - JENSEN, E. D. A distributed function computer for real-time control, em: Proceedings of the 1st Annual Symposium on Computer Architecture (1974), ACM, New York, NY, 176-182.

MOTO81 - MOTOROLA MICROPROCESSORS DATA MANUAL. Motorola Inc., Austin, Texas (1981), 4/818-4/835.

NATI84 - SERIES 32000 DATABOOK. National Semiconductor Corp., Santa Clara, CA (1984), 50-110 e 188-207.

PDSI83 - P&D SISTEMAS ELETRÔNICOS S/A. Especificações funcionais para o Sistema TELESP. P&D Sistemas Eletrônicos S/A, Campinas SP Brasil (1983).

PHIL84 - PHILLIPS, D. Memory-management varieties suit different application areas. EDN (6/Set/1984), 135-143.

PRAS70 - PRASSER, M. Interrupt-Organisation bei Realzeit-Datenverarbeitugsanlagen, em: Lecture Notes in Computer Science 13, Rechnerstrukturen und Betriebsprogrammierung (1970), Erlangen, 281-293.

SMIT82 - SMITH, A. J. Cache Memories. Computing Surveys 14, 3 (Set/1982), 473-530.

SPRY81 - SPRY, L. W. The management of extended memory for concurrent processes in T.I. microprocessor Pascal, em: Wescon/81 Electronic Show & Convention (15-17/Set/1981), Electronic Conventions, Inc., San Francisco, CA, session 9/1, 1-10.

UCRC85 - COMPONENTES DE SOFTWARE DO SISTEMA SITASU - UCR. P&D Sistemas Eletrônicos S/A, Campinas, SP, 1985.

USCE85 - COMPONENTES DE SOFTWARE DO SISTEMA SITASU - USCE. P&D Sistemas Eletrônicos S/A, Campinas, SP, 1985.

Z80D81 - Z80 DMA TECHNICAL MANUAL. Zilog, Inc., Cupertino, CA (1981).

ZILO81 - ZILOG. 1981 Data Book. Zilog, Inc., Corporate Communications, Cupertino, CA, 1981, 5-26 e 59-86.

CARACTERÍSTICAS GRÁFICAS

texto : - ambiente utilizado: microcomputador SME multiusuário - MP/M

- editor: VEDIT

- formatador: TEXT

- impressora: RIMA IS-1320 (SISTEMA)

figuras (exceto 2.7 e 5.4):

- ambiente utilizado: microcomputador MICROTEC XT2002 - CCP/M (compatível com IBM-PC)

- editor: software AUTOMIC

- impressora: GRAFIX MX-80