ELEMENTOS DO SISTEMA DE COMPUTAÇÃO

1.1 Arquitetura de Von Neumman

A Figura 1 reapresenta a máquina de Von Neumman cujo detalhamento foi apresentado nas primeiras sessões do curso.

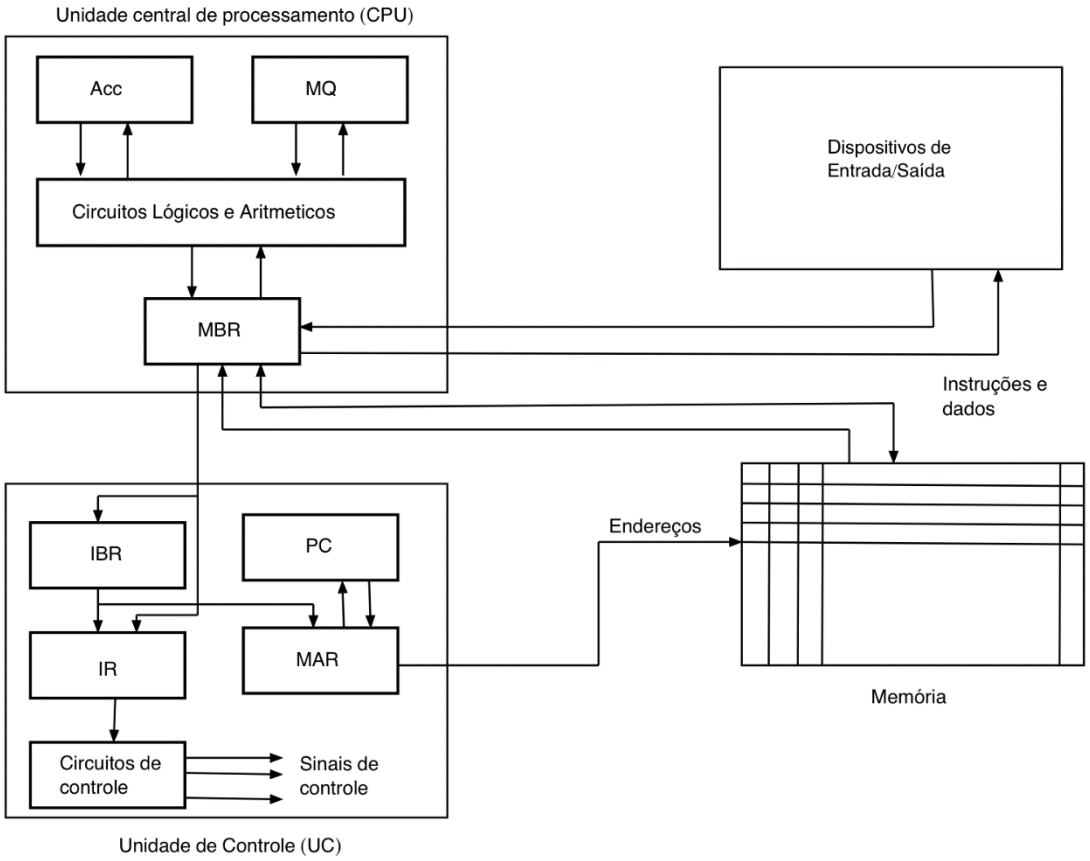


Figura 1 – Máquina IAS

A CPU (do inglês, *Central Processing Unit*), simplesmente, processador, é o elemento central da arquitetura. Contém dois elementos principais, Unidade de Controle e Unidade Lógica e Aritmética. Contém, ainda, elementos de armazenamento de dados, que podem ser registradores, podendo ser de uso geral ou com finalidade específica) ou uma memória interna cache.

Os elementos são interligados através de barramentos pelos quais trafegam os bits trocados entre eles. Os barramentos carregam os endereços dos dados acessados uma operação da CPU – barramento de endereços;

carregam os dados propriamente ditos; ou sinais de controle, que regulam o funcionamento dos elementos.

Externos à CPU existem a Memória Principal (MP) e os dispositivos de entrada e saída (*I/O*). Os dados da MP podem ser acessados e ocupam uma faixa de endereços que podem ser apontados pela CPU, por meio do barramento de endereços. O barramento de dados permite o tráfego bidirecional entre a MP e a CPU, especificamente com os registradores internos. Similarmente, bits de dados e endereços trafegam pelos respectivos barramentos entre a CPU e os variados dispositivos de *I/O*.

Na troca de bits, os elementos podem tomar a iniciativa de iniciar a transferência de dados ao barramento, enquanto outros aguardam as requisições. Os elementos ativos, que iniciam o processo de transferência de dados, são chamados de *masters* (mestres) e os elementos passivos, que esperam a requisição de transferência, são chamados de *slaves* (escravos). O comportamento de *master* ou *slave* não é fixo. Varia durante a operação da máquina.

1.2 MEMÓRIA

1.2.1 Hierarquia da memória

A estratégia de armazenamento de dados nos sistemas de computação pode ser resumida conforme apresenta a Figura 2.

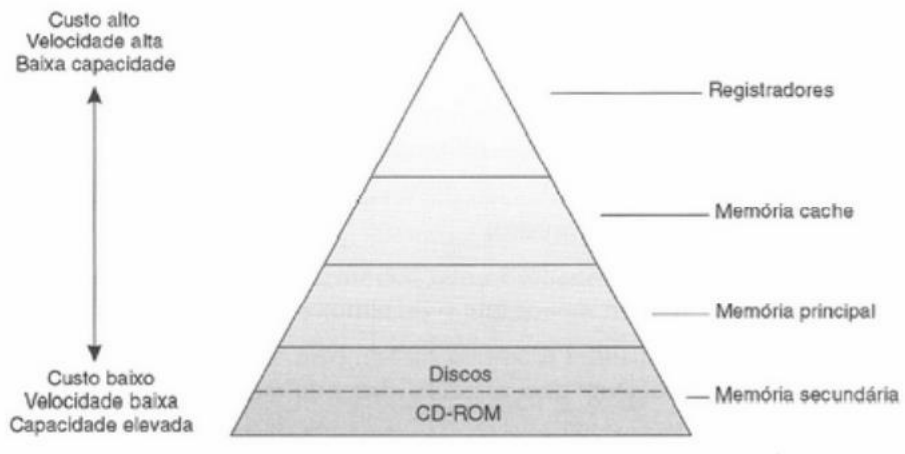


Figura 2 – Hierarquia da memória dos sistemas de computação

As características de performance de cada tipo de memória usado no sistema de computação podem ser representadas por uma pirâmide, que representa a capacidade de armazenamento de bits, a velocidade e o custo. Quanto mais o tipo se aproxima da base, maior é a capacidade de armazenamento e o tempo de uso e menor é o custo por bit armazenado.

Os quatro tipos de memória usados no sistema de computação são: memória secundária, memória principal, memória cache e registradores. Todos são dispositivos de armazenamento de dados e sua distribuição na arquitetura depende das características mencionadas acima.

A tabela da Figura 3 apresenta de forma mais precisa as características descritas de forma genérica na pirâmide anterior.

Tipo	Tempo de Acesso	Tamanho	Custo (por MB)
Registradores	Ciclos CPU	32-64 bits	---
L1	Ciclos CPU	32-64 Kbytes	---
L2	8-35 ns	512Kbytes -2 Mbytes	50 Us$
principal	40-120 ns	64 Mbytes – 1 Gbyte	1 Us$
secundária	5 ms	6 Gbytes-128 Gbytes	0.02 Us$

Figura 3 – Características dos tipos de memória

A capacidade armazenamento de memórias é um atributo que já foi tratado anteriormente. É definida como o número de bits que a memória pode armazenar. Pode ser definida em bytes (B), lembrando que cada byte possui 8 bits. Além disso, pode-se referir aos múltiplos de bits ou bytes, lembrando que 2^{10} representa 1 K, 2^{20} representa 1 M, 2^{30} representa 1 G.

A velocidade pode ser definida pelo tempo de acesso. O tempo de acesso representa quanto tempo a memória gasta para disponibilizar um dado após o endereço ser disponibilizado. O tempo de acesso depende da tecnologia de construção. Outro parâmetro que pode definir a velocidade da memória é chamado de ciclo de memória. O ciclo de memória é o tempo decorrido entre duas operações sucessivas de acesso à memória, seja para leitura ou escrita. Nesse sentido, o ciclo de memória compreende o tempo de acesso mais o tempo necessário para a realização de outras operações do sistema, logo $t_C = t_A + t_S$.

1.2.2 Tecnologias de construção

As características dos tipos de memória dependem fortemente da tecnologia de construção utilizada. A Figura 4 apresenta as diversas tecnologias existentes no mercado.

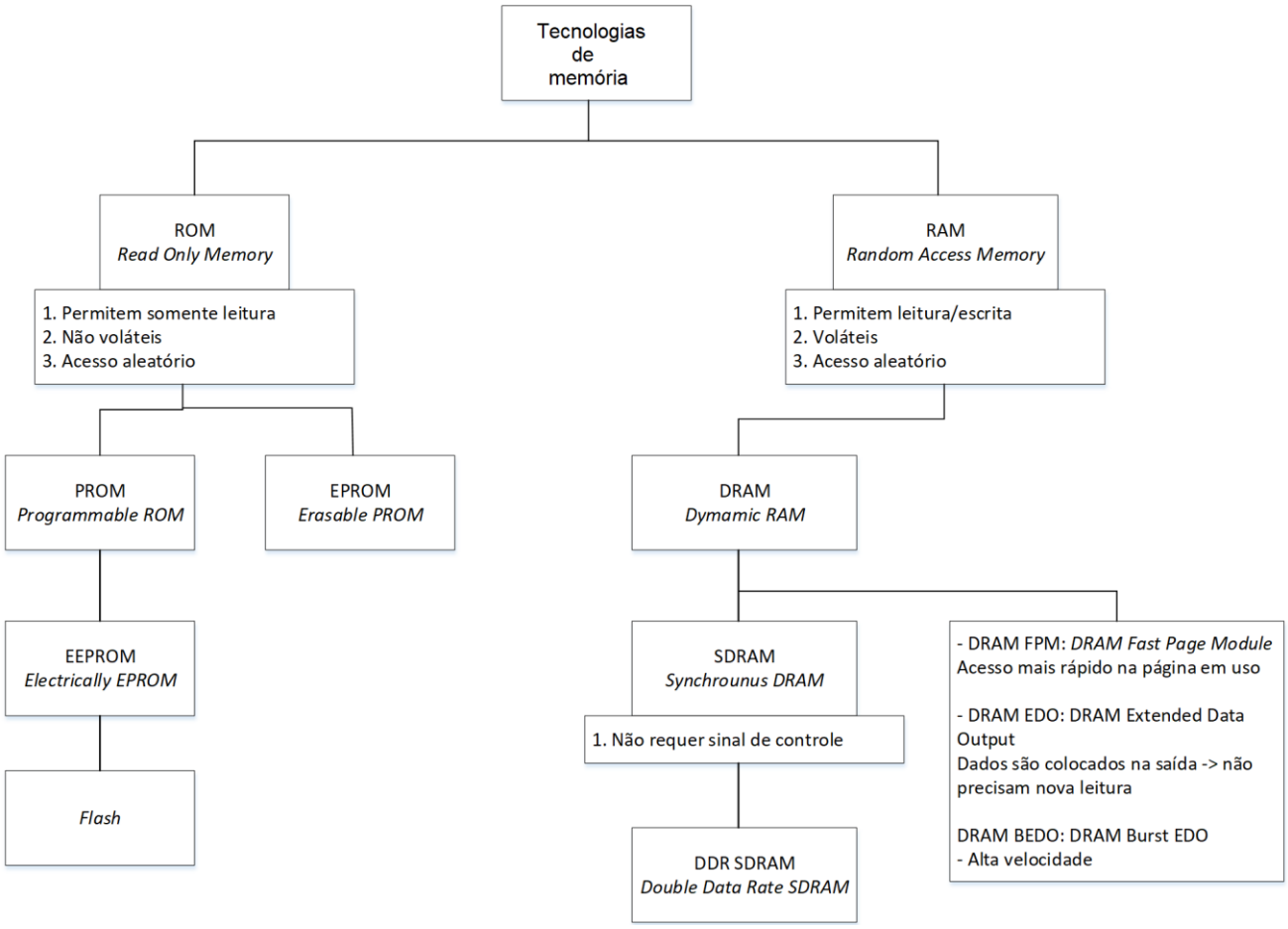


Figura 4 – Tecnologias de construção de memórias

A árvore de classificação distingue claramente duas tecnologias: ROM e RAM.

Quanto à forma de acesso, o acesso aleatório da construção de memória relaciona-se com o tempo de acesso curto e igual para todas as células. Nas memórias de acesso aleatório, não há necessidade de ler outras posições antes de acessar uma célula. Tempo de acesso curto e igual para todas as células. Existem tecnologias em que o acesso a uma célula requer uma sucessão preliminar de posições antes de acessar a posição de interesse. Nesse caso a memória é de acesso sequencial.

As memórias de acesso aleatório (RAM, do inglês *Random Access Memory*) admitem ser lidas ou ser escritas e mantêm os dados armazenados somente se houver alimentação elétrica. Essa característica define a volatilidade do armazenamento de dados. Memórias RAM são voláteis. As memórias somente de acesso (ROM, do inglês *Read Only Memory*) não admitem ser lidas e escritas livremente, com as RAMs. A escrita dos dados é realizada de acordo com condições controladas. Além disso, quanto à volatilidade, os dados armazenados na ROM permanecem mesmo na ausência de alimentação, portanto são tipicamente não voláteis.

Os sistemas de computação prescindem também do uso de memórias que sejam não voláteis, ou seja, que mantenham os dados armazenados mesmo quando haja ausência de alimentação elétrica. Essas memórias são chamadas de memória somente de leitura (ROM, do inglês *Read Only Memory*). O principal inconveniente das memórias ROM, qual seja, não admitir nova programação após a primeira ter sido feita, foi progressivamente sendo vencido pelo avanço tecnológico, com o surgimento de memórias que suportavam ser programadas conforme o interesse, embora somente uma vez (PROM, do inglês *Programmable* ROM), apagadas (EPROM, do inglês *Erasable* PROM) e reprogramadas (EEPROM, do inlgês *Electrically-Erasable* PROM). Mais recentemente, a memória *flash* é um tipo de EEPROM que pode ser reprogramada no próprio circuito, sendo cada vez mais utilizada nos sistemas de computação. As ROMs têm as seguintes aplicações típicas nos sistemas de computação:

- *Firmware* – armazenamento para inicialização das máquinas, integra hardware+software;
- *Bootstrap* – memória que armazena código de inicialização para chamar o sistema operacional;
- Tabela de dados – armazenar dados não voláteis;
- Conversor de códigos;
- Gerador de funções;
- Armazenamento auxiliar.

O tipo de armazenamento pode ainda ser realizado de duas formas: uma vez inserido o dado, ele permanece armazenado, o que define a tecnologia de memória estática; ou uma vez inserido o ano, para que ele se mantenha armazenado é necessário periodicamente uma recarga (*refresh*). Assim, as RAMs podem ser estáticas ou, por outro lado, podem ser construídas de forma que o sinal armazenado (1 ou 0) se mantém somente se houver um *refresh* (recarga) do valor armazenado em intervalos de tempo. Nesse caso são chamadas de RAMs dinâmicas.

As memórias RAM mais antigas operavam de modo que somente após a um sinal de controle do processador as fazia liberar os dados. Essa forma de operação caracteriza uma memória como assíncrona. Algumas tecnologias de memória assíncrona se sucederam e foram superadas por memórias síncronas SDRAM (DRAM síncrona). A aplicação de SDRAM elimina a necessidade de sinais de controle. A operação da memória acompanha a velocidade do clock do sistema, em cada transição do ciclo de clock. Nessa evolução, a mais recente é a memória SDRAM DDR (DDR, do inglês *Double Data Rate*), que admite a transferência do dado da memória para o barramento na transição positiva quanto na transição negativa do ciclo de *clock*.

A construção dos dispositivos de memória pode ser feita com semicondutores. São duas tecnologias muito utilizadas: MOS, semicondutor de óxido metálico e CMOS, semicondutor de óxido complementar. Além disso, podem ser construídas com meios magnéticos e meios ópticos.

Considerando as tecnologias de construção e voltando à hierarquia do sistema de memória na arquitetura pode-se definir que:

- As memórias secundárias são construídas em meios ópticos e magnéticos e sua função é armazenar grandes quantidades de dados, com a desvantagem de possuir tempo de acesso e de ciclo lentos, porém a um custo baixo;
- A memória principal é construída em semicondutor, com tecnologia de RAMs estáticas ou dinâmicas, possui média capacidade de armazenamento e custo. Sua principal função é armazenar os dados do programa de máquina e os dados armazenados a serem acessados pelo processador.

- A memória cache é construída em semicondutor, internamente ao próprio processador ou não, com tecnologia ROM, média/baixa capacidade de armazenamento, custo elevado. Sua principal função é armazenar os dados do programa de máquina e os dados armazenados local e temporalmente mais prováveis de serem acessados pelo processador. Utiliza o chamado princípio de localidade.
- Os registradores possuem capacidade de armazenamento de dezenas de unidade de bits, são construídos em dispositivos semicondutores e localizam-se internamente ao processador.

1.2.3 Memória Principal

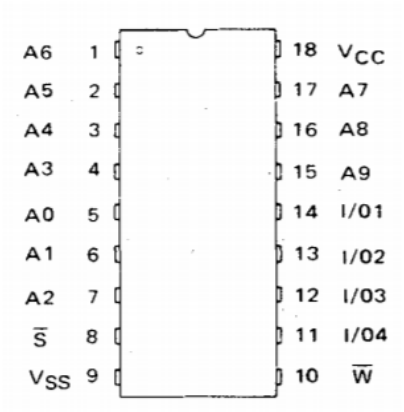
A MP armazena programas e dados. Os programas executados pela CPU contemplam um conjunto de instruções no nível do *set* de instruções do processador. As instruções compreendem normalmente um campo que define o seu código (código da instrução, chamado de *opcode*), que é definido pelo fabricante do processador e um campo dos operandos, bits sobre os quais será executada a instrução (existem vários formatos diferentes, embora esse seja o mais comum). As instruções são armazenadas na MP, ocupando um endereço bem definido. Assim como as instruções do programa, os dados operados pela CPU também ocupam um endereço da MP.

É fácil compreender o endereço de memória quando a entende-se como um conjunto de células, cada uma das quais armazenando 0 ou 1, organizadas como uma matriz. Cada linha corresponderá a 1 endereço específico. Em cada linha estão armazenados os dados, cada um dos quais pode ser lido/escrito naquele endereço – naquela linha da matriz. A MP contém M linhas de endereço de palavra de N bits. Os endereços podem ser apontados na forma binária por 2^L, onde L é o número de bits do endereço, tal que $2^L=M$. Em cada posição de endereço a MP armazena N bits.

Por exemplo, seja a instrução ADD Rd, Rr, uma instrução do nível ISA do processador ATmega2560 (será discutido detalhadamente mais à frente). Após executar essa instrução o processador soma os bits armazenados em dois registradores, Rd e Rr, carregando o resultado no registrador Rd. A instrução

ADD Rd, Rr possui 16 bits. O processador Atmega2560 admite duas instruções de 16 bits na MP por linha, logo pode operar com N=32=2^5 bits. Além disso, o Atmega2560 utiliza 16 bits para endereçamento de dados, ou seja, o barramento de endereços pode apontar para 2^{16} posições distintas. Se todos os dados pudessem ser armazenados na MP, a memória teria 2^5 x 2^{16} = 2^{21} bits, ou seja, 2 Mbits = 256 kB de capacidade de armazenamento.

O funcionamento da memória e como são armazenados os bits pode ser melhor entendido se explorada a Figura 5, que apresenta a memória RAM (*Random Access Memory*) do fabricante Motorola, que possui capacidade de armazenar 4096 bits ou seja 4kbits.



(a)

PIN NAMES	
A0-A9	Address Input
$\overline{W}$	Write Enable
$\overline{S}$	Chip Select
I/O1-I/O4	Data Input/Output
V_{CC}	Power (+5 V)
V_{SS}	Ground

(b)

Figura 5 – Memória Motorola MCM2114 de 4096 bits (a) Pinagem (b) Identificação dos pinos

Os pinos A0-A9 recebem os bits do barramento de endereço, portanto 10 linhas de bits, sendo o menos significativo A0 e o mais significativo A9. Como o barramento de endereços entrega 10 bits, é possível armazenar em 2^{10}, ou seja, 1024 posições distintas de memória, na faixa 000H a 3FFH (hexadecimal). Os pinos I/O1-I/O4 recebem os bits do barramento de dados, ou seja, em cada posição de memória é possível ler/escrever uma palavra de 4 bits. O pino $\overline{W}$ habilita leitura ou escrita na memória (a memória RAM possiblita leitura ou escrita dos dados armazenados). Se $\overline{W}$=0, os dados do barramento de dados serão escritos na posição de endereço indicada em A0-A9. Em caso contrário, os dados serão lidos daquela posição. O pino $\overline{S}$ habilita a operação ou desabilitação

da memória. Se $\bar{S}$=0 a memória está habilitada para ser usada. Trata-se de um sinal de controle do chip. Os pinos V_{CC} e V_{SS} são alimentação do chip.

Cada linha da matriz corresponde a uma célula de memória. Células adjacentes em memória têm endereço consecutivo. O tamanho típico de cada célula de memória atualmente é de 8 bits (1 byte). Em geral, os bytes são agrupados em palavras de 4 bytes ou 8 bytes.

As tecnologias das memórias têm evoluído de forma bastante rápida, de modo especialmente que o tempo de espera para ler um dado ou escrever um dado na memória não prejudique o desempenho global do sistema. Por exemplo, a memória Motorola MCM2114 possui tempo de ciclo de leitura fixo de 200 nanosegundos, nesse sentido entre a requisição do dado pelo processador e a liberação do dado no barramento serão consumidos 200 ns.

1.2.4 Memória cache

Considerando a hierarquia do sistema de memória discutido anteriormente, a memória cache ocupa o terceiro nível a partir da base da pirâmide. Tipicamente, possui média/baixa capacidade armazenamento, custo alto, porém velocidade elevada. Essas características a tornam muito útil no equilíbrio custo benefício no desempenho da arquitetura.

O princípio que permite a utilização da cache se chama princípio da localidade, segundo o qual os programas armazenados que controlam a execução da máquina se situam localmente próximos no tempo ou no espaço. A proximidade de localidade espacial diz respeito ao endereço das instruções executadas. Normalmente quando uma instrução é executada as próximas instruções estão em endereços de memória próximos (lembre-se que, no nível ISA, o contador de programa (PC) tipicamente sofre incrementos de uma unidade, violando essa condição somente quando instruções de desvio devem ser executadas). Além dessa localidade espacial próxima, a localidade temporal se caracteriza pelo fato de que uma instrução de um código no nível ISA tem a probabilidade elevada de voltar a ser executada novamente em um curto espaço de tempo após ter sido executada em um determinado instante. Observe que o princípio de localidade está diretamente relacionado à probabilidade de uma

instrução ser executada novamente após ter sido executada, seja por sua localização no endereço, seja pelo tempo transcorrido desde que foi executada.

A Figura 6 apresenta um sistema de memória em que se utiliza três níveis de cache: uma cache interna ao encapsulamento do processador, chamada cache de nível 1 (L1); uma cache no pacote da CPU – nos circuitos diretamente ligados ao processador, de nível 2 (L2); e uma cache externa de nível 3 (L3).

No nível L1, instruções e dados do programa de máquina são separados. No nível L2, instruções e dados do nível L3 são armazenados de modo a estarem prontamente disponíveis para leitura/escrita via barramento local, desonerando o desempenho do tempo devido à transmissão no barramento externo. No nível L3 instruções e dados são armazenados, desonerando o desempenho do tempo de resposta da memória principal (RAM, mais lenta do que a cache).

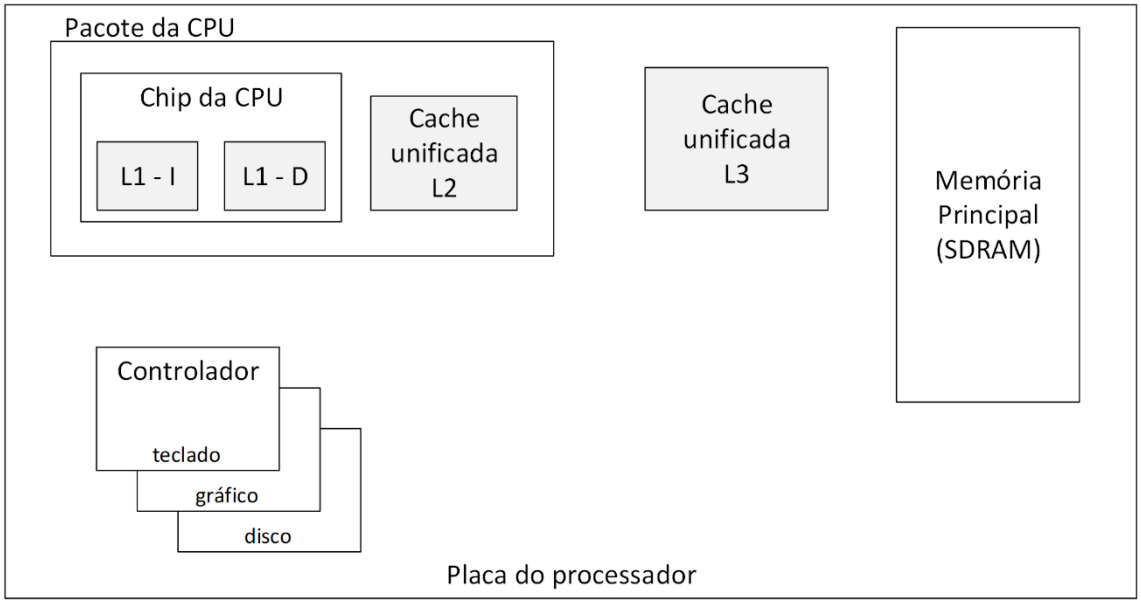


Figura 6 – Arquitetura de memória com cache de três níveis

Como a arquitetura explora na operação o princípio de localidade em favor do aumento de desempenho? Vamos tratar especificamente do nível L3 e generalizar a operação para os demais níveis.

Para controlar a operação da cache é necessário utilizar um circuito chamado controlador de cache, que produz sinais de controle de sincronização dos procedimentos de leitura e dados do processador no sistema de memória. A Figura 7 destaca o controlador.

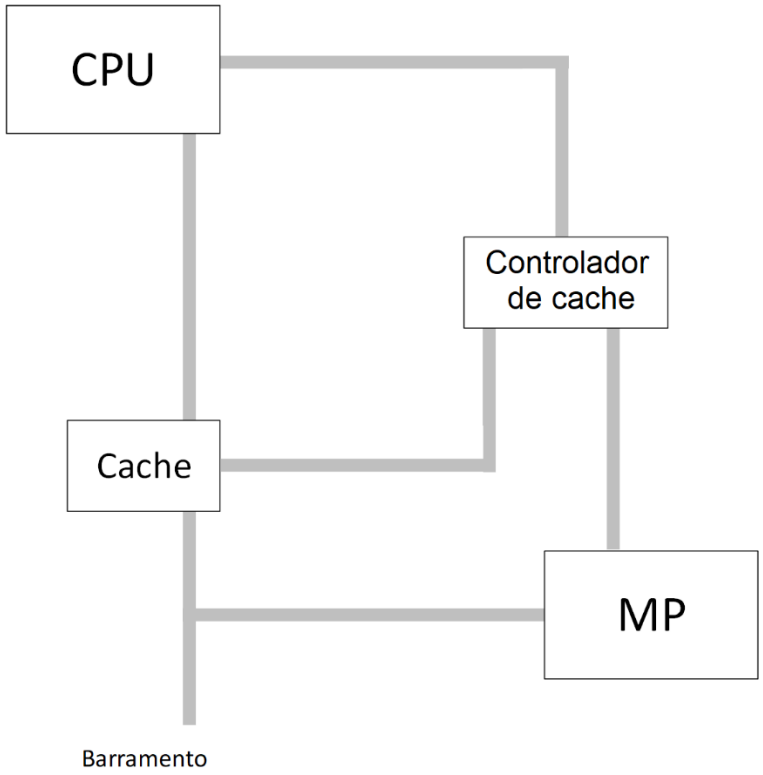


Figura 7 – O controlador de cache

A Figura 8 ilustra a organização da memória principal para a exploração do princípio de localidade na leitura/escrita de dados. A MP é dividida em blocos de tamanho fixo chamados linhas de cache. Os blocos são armazenados na cache. Em qualquer instante alguns blocos estão na cache. Na cache os blocos da MP armazenados são identificados.

End MP	Linha	Dado
0	0	
1		
2		
3		
4	1	
5		
6		
7		
8	2	
9		
10		
11		
12	3	
13		
14		
15		
16	4	
17		
18		
19		
20	5	
21		
22		
23		
24	6	
25		
26		
27		
28	7	
29		
30		
31		
32	8	
33		
34		
35		
36	9	
37		
38		
39		
40	10	
41		
42		
43		
44	11	
45		
46		
47		

Figura 8 – Organização da MP em N=12 linhas de cache, N.K=12.4 linhas de endereço

Em síntese, a cache é dividida em M blocos de tamanho fixo com K linhas de endereço, cada uma armazenando uma palavra, portanto totalizando M.K linhas de endereço. A MP é dividida em N linhas de cache com capacidade de armazenar as K linhas de dados de um bloco da cache, logo a MP possui N.K linhas de endereço.

Como tratado anteriormente, os endereços da MP são definidos de 0 a N.K-1; as linhas de cache são numeradas e divididas de 0 a N-1 – cada linha de cache define uma faixa de endereços da MP. A cache é dividida em blocos numerados a partir de 0 até M-1. A Figura 9 ilustra o relacionamento das linhas de cache da MP com os blocos da cache.

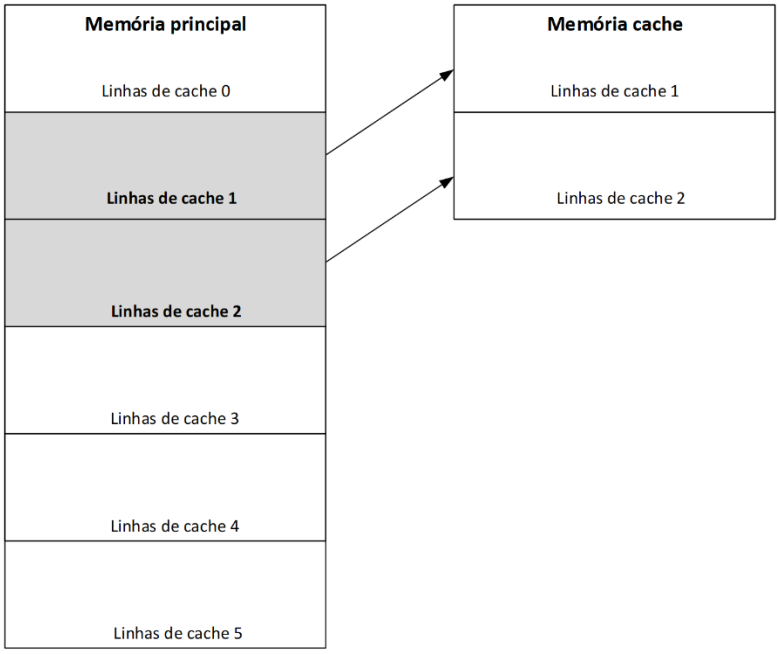


Figura 9 – Relacionamento N=6 linhas de cache da MP com M=2 blocos da cache

A capacidade de armazenamento da MP é maior do que a cache, consequentemente N>M, ou seja, nem todos dados da MP podem ser mapeados na cache. A utilização da cache na arquitetura requer a aplicação de um processo, que está apresentado na Figura 10.

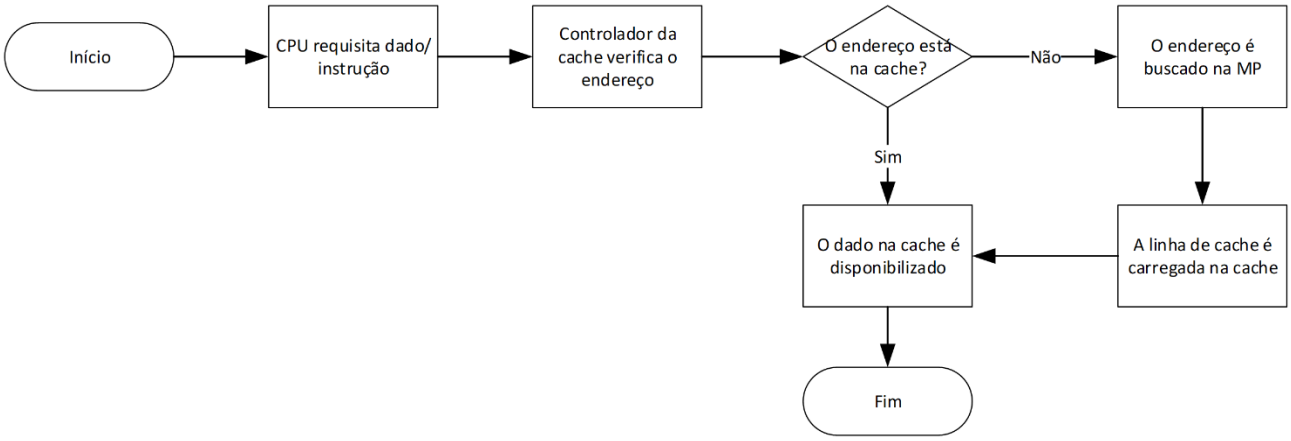


Figura 10 – Processo de atendimento à requisição de acesso ao sistema de memória com o uso da cache executado pelo controlador de cache

Na execução de um programa o processador requisita um dado ou instrução armazenado em um endereço de dados da MP. A requisição é realizada a um circuito chamado de controlador da cache. O controlador verifica inicialmente se o dado do endereço requisitado está presente na cache, em qual bloco e endereço da cache se encontra. Caso esteja presente, configura-se o que se chama de acerto. O dado/instrução é acessado com o tempo de resposta da cache; em caso contrário, caracteriza-se o que se chama de falta e o dado/instrução é buscado no endereço da MP, com o tempo de resposta da MP, atualizado na cache e disponibilizado para o processador.

A eficácia do uso desse mecanismo pode ser facilmente determinada pelo tempo médio de acesso à memória com e sem a utilização da cache. É necessário definir o tempo médio de acesso ao sistema de memória, calculado por $t_M = c + (1 - h).m$, onde *c* é o tempo de acesso à cache; *m* é o tempo de acesso à MP; e *h* é a taxa de acertos. A taxa de acertos pode ser calculada admitindo que em k acessos à memória ocorreu k-1 acertos e 1 falta, logo a taxa de acertos é a razão (k-1)/k.

Para o processo descrito acima ser executado é necessário definir uma política de mapeamento dos dados da memória primária na cache que objetive dizer quais blocos da cache serão copiados na MP. Existem três técnicas de mapeamento: mapeamento direto; mapeamento totalmente associativo; e mapeamento associativo por conjunto.

No mapeamento direto, o relacionamento é predefinido de linhas de cache da MP com os blocos da cache. O controlador de cache possui uma tabela que estabelece a associação, conforme apresenta a Figura 11.

Na tabela está evidenciado que a MP possui N=48 linhas de endereço, 12 linhas de cache. A cache possui M=3 blocos de K=4 palavras. No exemplo, as linhas de cache 3n, n=0 a 3, estão mapeados no bloco 0; as linhas de cache 3n+1, n=0 a 3, estão mapeados na linha 1; e as linhas de cache 3n+2, n=0 a 3, estão mapeados na linha 2. Os endereços da MP 0,1,2,3 pertencem à linha de cache 0 na MP e estão mapeados nos respectivos endereços 0,1, 2 e 3 do bloco 0 da cache; os endereços da MP 4, 5, 6 e 7 pertencem à linha de cache 1 na MP e estão mapeados nos respectivos endereços 4, 5, 6 e 7 da cache, que correspondem aos endereços 0, 1, 2 e 3 do bloco 1.

Para o controlador de cache executar o algoritmo apresentado na Figura 10, o início é a requisição pela CPU da instrução ou dado em um endereço da MP. O controlador verifica qual é a linha de cache que armazena o dado e em qual bloco o dado ou instrução poderia estar mapeado.

Para verificar qual é a linha *L* que armazena o dado ou instrução pode-se aplicar

$L = X\ Div\ K$, onde K é o número de posições de cada linha de cache na MP.

A posição relativa *R* do endereço *X* da MP na linha de cache na MP, consequentemente no correspondente bloco da cache, é dado por

$R = X\ Mod\ K$.

Para verificar qual é o bloco *B* que armazenaria aquela linha, o controlador deve aplicar

$B = L\ Mod\ M$, onde M é o número de blocos da cache.

End MP	Linha	Cache	End cache	Dado
0	0	0	0	
1			1	
2			2	
3			3	
4	1	1	4	
5			5	
6			6	
7			7	
8	2	2	8	
9			9	
10			10	
11			11	
12	3	0	0	
13			1	
14			2	
15			3	
16	4	1	4	
17			5	
18			6	
19			7	
20	5	2	8	
21			9	
22			10	
23			11	
24	6	0	0	
25			1	
26			2	
27			3	
28	7	1	4	
29			5	
30			6	
31			7	
32	8	2	8	
33			9	
34			10	
35			11	
36	9	0	0	
37			1	
38			2	
39			3	
40	10	1	4	
41			5	
42			6	
43			7	
44	11	2	8	
45			9	
46			10	
47			11	

Figura 11 – Tabela de mapeamento direto MP-cache

A cache é organizada de modo que: cada bloco possui um campo com o conjunto de dados (*data*) do tamanho da linha de cache da MP; cada bloco possui um campo com um *flag* indicando se os dados do bloco são válidos ou não, isto é, informando se os dados que precisam ser lidos estão presentes na cache; e possui um campo que descreve qual é a linha de cache da MP que está armazenada no campo de dados (campo *tag*). Com essas informações de

entrada da cache é possível o controlador implementar a leitura dos dados do sistema de memória seja da cache seja da MP, otimizando o desempenho global do sistema. A técnica de mapeamento direto aumenta a eficiência do sistema, entretanto possui o inconveniente de manter dados armazenados na cache mesmo quando não são muito utilizados em virtude do relacionamento pré-determinado linhas de cache da MP-blocos da cache.

Na técnica de mapeamento totalmente associativo, não há uma relação pré-determinada de linhas de cache com blocos, o que implica aumento de eficiência. No entanto, aumenta-se a complexidade do circuito necessário à implementação do processo de espelhamento de dados. O mapeamento associativo por conjunto concilia a simplicidade do mapeamento direto com a eficiência do totalmente associativo.

No mapeamento dos dados da MP para a cache, uma vez que o dado de um determinado endereço não esteja presente na cache, ou seja, tendo ocrrido uma falta, é necessário buscar o dado na memória e substituir os dados de algum dos blocos da cache. Existem algumas técnicas que o controlador pode utilizar como: FIFO – *First-In-First-Out*, em que o bloco cujos dados foram os primeiros a ser inseridos são aqueles a ser substituídos; LRU – *Least Recently Used*, em que o bloco que há mais tempo não é acessado deve ser substituído.

O controlador de cache não opera somente com a implementação do processo de atendimento à requisição de leitura de dados pelo processador. O controlador também deve se encarregar de atualizar os dados escritos na cache para a MP. Existem algumas técnicas possíveis de implementar a atualização dos dados da cache na MP, como, por exemplo, a escrita direta (*write through*) e a escrita retroativa ou retardada (*write back* ou *write deferred*).

Na escrita direta, ilustrada na Figura 12, quando se escreve um dado na cache o controlador simultaneamente escreve os dados também na MP. Dessa forma os dados de uma linha de cache da MP sempre são iguais aos do bloco da cache. Isso implica perda de eficiência em virtude do tempo necessário para realizar a escrita na MP. Uma forma mais eficiente de realizar a substituição é adiar para realizá-la somente quando um bloco da cache precisar ser substituído

por um novo, uma vez que tenha se configurado uma falta. A Figura 13 ilustra essa operação.

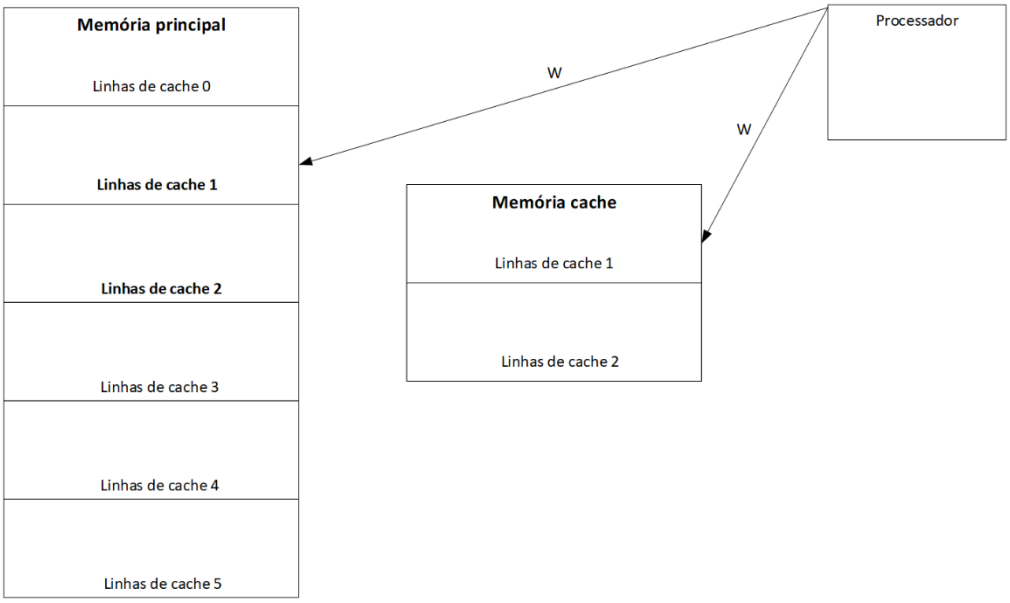


Figura 12 – Atualização de dados da MP por escrita direta

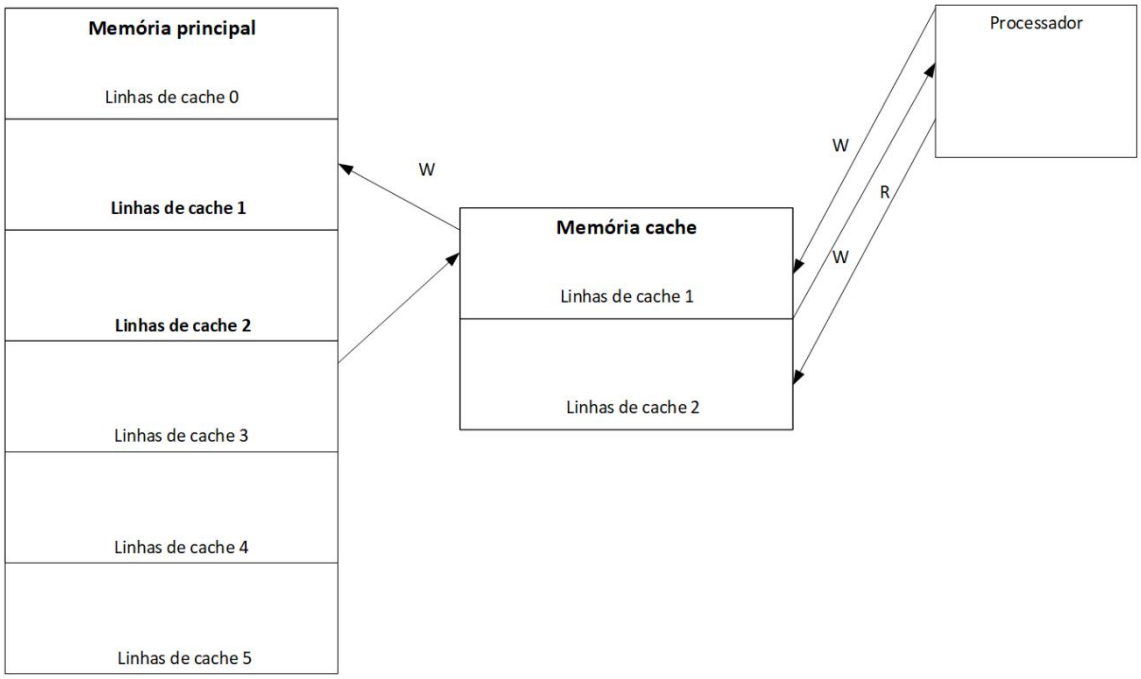


Figura 13 – Atualização de dados da MP por escrita retardada

1.2.5 Registradores

Além da memória primária e da cache, são dispositivos utilizados para armazenamento de dados importantes no sistema computacional os

registradores. A CPU pode ter *k* registradores (R_1 a R_k), alguns de uso geral, outros de uso específico. Dependendo da sua finalidade, os registradores possuem uma capacidade específica de armazenamento de bits.

Alguns registradores de uso específico normalmente são encontrados nas arquiteturas. Por exemplo, o Contador de Programa (PC, do inglês *Program Counter*), que se destina a armazenar o endereço em que a próxima instrução ISA de um programa armazenado na memória se localiza; e o Registrador de Instrução (IR, do inglês *Instruction Register*), que se destina a armazenar a instrução ISA de um programa armazenado na memória e buscada pela CPU.

REFERÊNCIAS

ATMEL. **Datasheet Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V**. [S.l.]: [s.n.]. ISBN 2549Q–AVR–02/2014.

FLETCHER, W. I. **Engineering Approach to Digital design**. [S.l.]: Prentice-Hall, 1980.

GOOGLE. Disponivel em: <www.google.com.br>. Acesso em: 27 abril 2020.

KERSCHBAUMER, R. **Microcontroladores**. [S.l.]: [s.n.], 201-. Disponivel em: <http://professor.luzerna.ifc.edu.br/ricardo-kerschbaumer/wp-content/uploads/sites/43/2018/02/Apostila-Microcontroladores.pdf>. Acesso em: 15 jul. 2020.

LOUDEN, K. C. **COMPILADORES princípios e práticas**. São Paulo: THOMSON, 2004.

MARTINS, G. O.; CAMARGO, J. T. F. D.; VERASZTO, E. V. SIMBLER – UM SIMULADOR DE LINGUAGEM DE MONTAGEM DIDÁTICO APLICADO AO ENSINO DE INFORMÁTICA. **Interciência e Sociedade**, p. 118-128.

MICROCHIP TECHNOLOY LTDA. **AVR® Instruction Set Manual**. [S.l.]: [s.n.]. ISBN DS40002198A.

ROBOCORE. ROBOCORE. **ROBOCORE**. Disponivel em: <https://www.robocore.net/loja/#>. Acesso em: 2020.

SEBESTA, R. W. **Conceitos de Linguagens de Programação**. 4a. ed. Porto Alegre: Bookman, 2000.

TANENBAUM, A. S. **Organização estruturada de computadores**. 5a. ed. Rio de Janeiro: Prentice-Hall, 2007.

WIKIPÉDIA. Disponivel em: <pt.wikipedia.org/wiki/Linguagem>. Acesso em: 27 abril 2020.

ZUFFO, J. A. **SISTEMAS ELETRÔNICOS DIGITAIS organização interna e projeto vol 1**. 2ª. ed. [S.l.]: Edgar Blucher, 1976.