Arquitetura de Computadores

Para desenvolvedores

Antonio Carlos Souza
Igor Alexandre de Lima

IXTLAN

Arquitetura de Computadores

Para desenvolvedores

Antonio Carlos Souza

Igor Alexandre de Lima

São Paulo – 2018

Projeto gráfico:

Editora Ixtlan

Diagramação:

Márcia Todeschini

Capa:

Gabriel Polizello

Antonio Carlos Souza

Igor Alexandre de Lima

Arquitetura de computadores para desenvolvedores – São Paulo/SP - Ed. Ixtlan, Outubro/2018

ISBN: 978-85-8197-721-8

1. Ciência da computação 2. Desenvolvimento de softwares

CDD 000

Editora Ixtlan - CNPJ 11.042.574/0001-49 - I.E. 456166992117

We would like to thank God for all the blessings given during our journey.

Prefácio

Diversos livros trabalham de forma brilhante o tema e disciplina de Arquitetura de Computadores para inúmeros cursos de bacharelado ou engenharia da computação, ofertando aos centros de discussão várias possibilidates de explicações ou exemplos para apresentar aos discentes os conceitos da área. Para o contexto de um curso tecnológico de análise e desenvolvimento de sistemas, em geral, os alunos não tem acesso a disciplinas anteriores como sistemas digitais ou eletrônica.

O desejo de confeccção desse livro surge com o objetivo de reforçar os pontos considerados relevantes na carreira do futuro programador ou desenvolvedor relacionados à disciplina de arquitetura de computadores como:

- organizar os discursos e termos para formação do profissional;
- articular os elementos com o funcionamento de um sistema computacional;
- a fluencia em lógica de programação com base na arquitetura do computador.
- entender os gargalos do sistema computacional e prováveis soluções

Tudo isso sem avolumar o texto.

Abrangência

O primeiro destinatário da obra é o aluno do Curso Superior Tecnológico de Análise e Desenvolvimento de Sistemas do Instituto Federal da Bahia - IFBA campus Salvador.

Máquina de Glenn

No livro, abordaremos as rotinas do processador, usando a máquina descrita por Glenn. Herdando seu nome do autor do livro Ciência da Computação: uma visão abrangente, J. Glenn Brookshear, a máquina de Glenn é um sistema composto de diversos registradores e de uma memória principal

Sumário

Lista de Tabelas

Lista de Figuras

1

Introdução

No Curso Superior Tecnológico de Análise e Desenvolvimento de Sistemas, o discente que cursa a disciplina de Arquitetura de Computadores nem sempre tem um domínio de programação ou de lógica de programação ou está aprendendo no mesmo semestre.

Um cenário como esse pode ser uma ameaça ao bom aproveitamento e ao aprendizado ou uma oportunidade para aprender a programar na linguagem próxima a linguagem da máquina (0,1). Nesta obra, usaremos a máquina de Glenn, que é um sistema composto de diversos registradores e de uma memória principal, usando a rotina de busca, decodificação e execução do processador.

Depois de praticar e reforçar a lógica de progrmação, o aluno parte para entender como melhor usar as memórias e como impactam no sistema computacional, visando uma melhor performance do seu código.

Em um terceiro momento, o aluno pode entender como o sistema interno (processador e memória) faz a comunicação com dispositovs de entrada e saída. Os discos rígidos, RAID, Barramento e SSDs também fazem parte dos pontos importantes dessa troca de dados e seus gargalos.

Com a fluencia do funcionamento do processador, memórias e dispositivos de entrada e saída. O futuro programador precisa estudar os tópicos avançados como pipeline, simd, mimd, hyperthreading e cpu-multicore.

Por fim, para a visão de mais alto nível, do processo (programa em execução) e nao mais de instrução a instrução do programa (visão do processador), é apresentado o sistema operaciional multitarefa, os conceitos de processo, memória virtual são explicados. É sugerido o uso do Simulador SOSIM do Luiz Paulo Maia para a prática da execução de vários proces-

sos, visualizando o impacto na memória e memória virtual. Uma opção de simulador que igual ao SOSIM mostra a execução dos processos e uso da memória e, diferente do Simulador do Luiz Paulo Maia, apresenta a gerência de Disco é o Simulador SWSO, Simulador Web de SO, projeto de TCC de Ramon Almedia Oliveira, graduado pelo curso de ADS do IFBA campus Salvador.

2

Capítulo 2 - Conjunto de instruções

> *"You insist that there is something that a machine can't do. If you will tell me precisely what it is that a machine cannot do, then I can always make a machine which will do just that."*
>
> – John Von Neumann, *1948*

2.1 Começando por uma máquina simples

Computadores são máquinas compostas de hardware, a parte física, e software, a parte lógica, capazes de realizar processamentos de dados, armazenando e manipulando informações. Toda máquina, para ser considerada um computador, essencialmente tem de possuir uma Unidade Central de Processamento, mais conhecida como CPU, memória e dispositivos de entrada e saída.

A arquitetura de computadores é o nome dado a organização funcional dos diversos componentes e parâmetros de um sistema computacional. Ela que estabelece a qualidade e desempenho dos aspectos compreendidos pelo usuário.

Neste capítulo iremos abordar a máquina descrita por Glenn. Herdando seu nome do autor do livro Ciência da Computação: uma visão abrangente, J. Glenn Brookshear, a máquina de Glenn é um sistema composto de diversos registradores e de uma memória principal.

Registradores, ou apenas registros, são porções de memória localizadas no processador (CPU), mais precisamente no interior de microprocessadores, que são capazes de armazenar provisoriamente informações. Os regis-

tradores se encontram no mais alto grau da hierarquia de memória devido a sua excelente velocidade de transferência de dados que permite um rápido acesso aos mesmos, o que torna essa tecnologia excessivamente cara. Em razão disso, os registradores possuem baixa capacidade de armazenamentos, se encontrando apenas na casa dos bytes.

Os registradores são utilizados para guardar valores durante operações, isto é, quando a memória principal necessita armazenar dados para usá-los futuramente, os dados são deslocados para os registradores onde são executados pelo processador e armazenados até que sejam deslocados de volta para a memória principal.

Há também registradores de propósito específico como:

- O Program Counter (PC) ou contador de programa, que atua como um marcador, ele aponta a posição na memória onde a próxima instrução será processada, e;

- O Instruction Register (RI), que armazena a instrução a ser executada.

A memória principal, mais conhecida como memória RAM (Random Acess Memory), é a parte da memória imprescindível para o funcionamento do computador, dada sua ligação direta ao processador que faz uso dessa união para buscar instruções a serem executadas. Localizada na placa-mãe, sua memória é temporária devido à eliminação de seus dados perante o encerramento do computador.

2.1.1 A arquitetura da máquina

A máquina é composta de 16 registradores de propósito gerais, numerados de 0 a F (no sistema hexadecimal), cada um possuindo apenas um byte, ou 8 bits (2 hexadecimais), de comprimento. Para indicar os registradores nas instruções, cada registrador é associado unicamente a um padrão de quatro bits que representa o número do registrador correspondente. Assim, o registrador 0 é identificado por 0000 (hexadecimal 0), e o registrador 4 por 0100 (hexadecimal 4). O número dentro do registrador pode estar tanto em binário quando em hexadecimal.

A memória principal consiste de 256 células, cada qual com 8 bits de dados. Como existem 256 células na memória, a cada uma é associado univocamente um endereço que consiste de um inteiro num intervalo de 0 a 255. Um endereço pode ser, portanto, representado por um padrão de oito bits, que varia de 00000000 a 11111111 (ou um valor hexadecimal, no intervalo de 00 a FF).

2.1.2 A linguagem da máquina

Cada instrução da máquina possui dois bytes de comprimento, isto é, 16 bits ou duas células de memória. Os primeiros quatro bits constituem o código de operação enquanto os últimos doze bits compõem o campo do operando. A tabela seguinte contém a lista das instruções, chamada de ISA (Instruction Set. Architecture - Arquitetura do Conjunto de Instruções), juntamente com uma breve descrição de cada uma delas. As letras R, S e T são usadas nesses campos no lugar de dígitos hexadecimais para identificarem um registrador, que varia dependendo de cada particular aplicação da instrução. As letras X e Y são usadas no lugar de dígitos hexadecimais nos campos variáveis que não representam um registrador, elas são atribuições ou posições de memória.

Tabela 2.1: Linguagem da máquina

Código de operador	Registrador/Memória	Descrição
1	RXY	LOAD
2	RXY	LOAD
3	RXY	STORE
4	0RS	MOVE
5	RST	ADD
6	RST	SHIFT
7	RST	OR
8	RST	AND
9	RST	EXCLUSIVE
A	R0X	ROTATE
B	RXY	JUMP
C	000	HALT
D	0RS	PAINT

Códigos de operação

Para fins didáticos, omitiremos registradores, posições de memória e de pixels não utilizados em cada operação.

x	y	x OR y	x AND y	x XOR y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Figura 2.1: Imagem com OR, AND e OR EXCLUSIVE para auxílio

1) LOAD (carrega) o registrador R com o padrão de bits encontrado na

posição de memória de endereço XY.

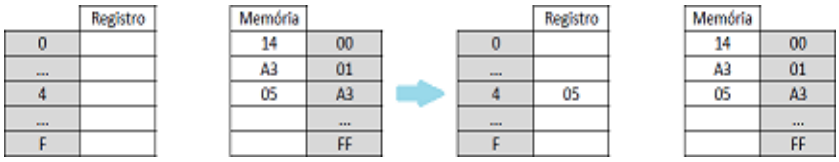


Figura 2.2: Operação 1

No exemplo acima, 14A3 carrega o conteúdo da posição de memória de endereço A3 no registrador 4.

2) LOAD (carrega) o registrador R com padrão de bits XY.

Figura 2.3: Operação 2

A instrução 28A3 coloca A3 no registrador 8.

3) STORE (armazena) o padrão de bits encontrado no registrador R na posição de memória de endereço XY.

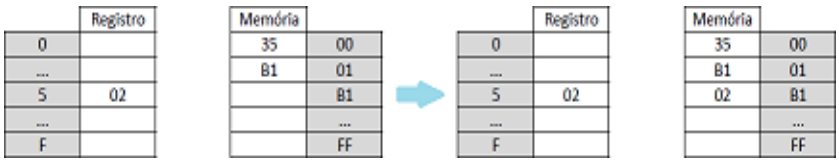


Figura 2.4: Operação 3

35B1 armazena o conteúdo do registrador 5 na posição de memória de endereço B1.

4) MOVE (copia) o padrão de bits encontrado no registrador R para o registrador S.

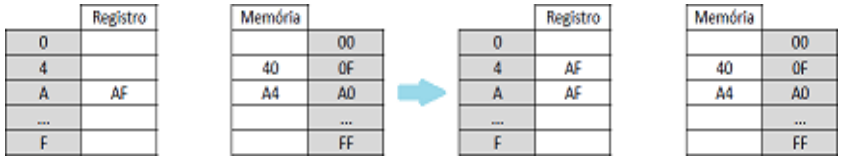


Figura 2.5: Operação 4

40A4 copia o conteúdo do registrador A para o registrador 4.

5) ADD (soma) os padrões de bits dos registradores S e T, em complemento de 2, e coloca o resultado no registrador R.

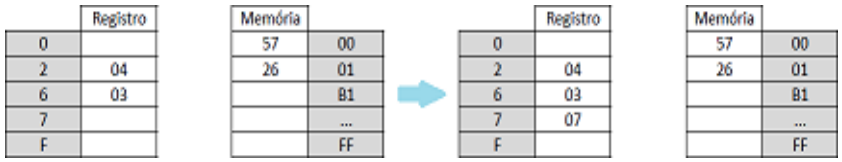


Figura 2.6: Operação 5

5726 soma os valores binários dos registradores 2 e 6 e coloca no registrador 7 esse resultado.

6) SHIFT (desloca) o padrão de bits do registrador R X bits para a direita. Deslocar apenas uma vez para a direita divide o padrão de bits por 2.

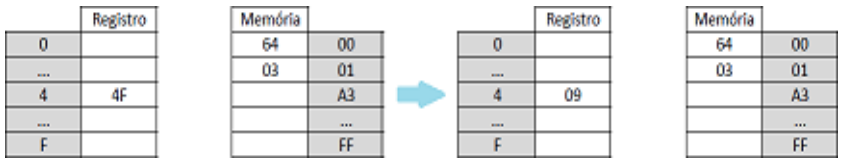


Figura 2.7: Operação 6

6403 desloca 3 bits para a direita do conteúdo do registrador 4, o conteúdo do registrador 4, 01001111, se transforma em 00001001.

7) OR (ou) executa operação lógica (ou) sobre os padrões de bits dos registradores S e T e coloca o resultado no registrador R.

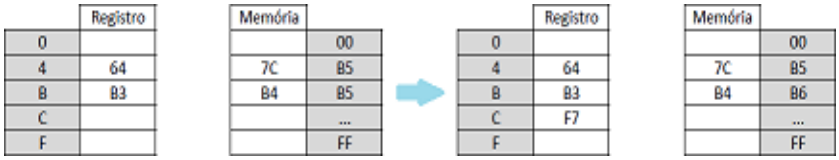


Figura 2.8: Operação 7

7CB4 coloca no registrador C o resultado da operação OR com os conteúdos dos registradores B e 4.

8) AND (e) executa a operação lógica (e) sobre os padrões de bits dos registradores S e T e coloca o resultado no registrador R.

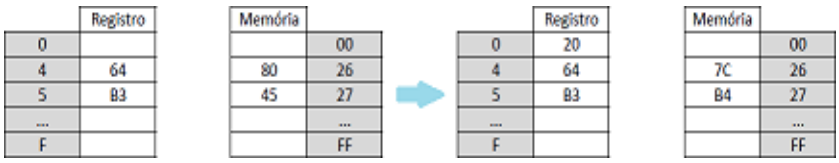


Figura 2.9: Operação 8

8045 coloca no registrador 0 o resultado da operação AND entre os conteúdos dos registradores 4 e 5.

9) EXCLUSIVE OR (ou exclusivo) executa operação OU EXCLUSIVO sobre os registradores S e T e coloca o resultado no registrador R.

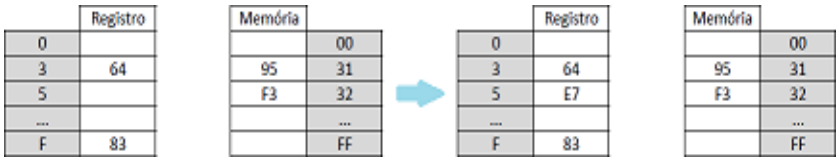


Figura 2.10: Operação 9

95F3 coloca no registrador 5 o resultado da operação entre EXCLUSIVE OR entre os conteúdos dos registradores F e 3.

A) ROTATE (gira) o padrão de bits do registrador R X bits para a direita. E em todas às vezes colocando o bit que está na extremidade inferior na extremidade superior.

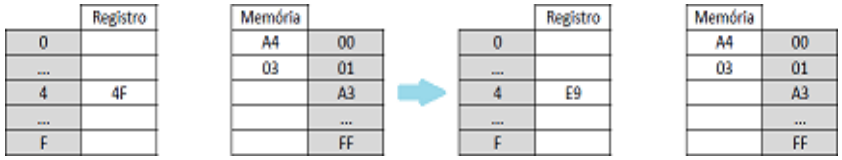


Figura 2.11: Operação A

A403 gira 3 bits para a direita do conteúdo do registrador 4 de forma circular, o conteúdo do registrador 4, 01001111, se transforma em 11101001 .

B) JUMP (salta) para a instrução localizada na posição de memória de endereço XY se o padrão de bits do registrador R coincidir com o padrão de bits do registrador 0. Caso contrário, prosseguir na seqüência normal da execução.

Caso o padrão de bits coincidir:

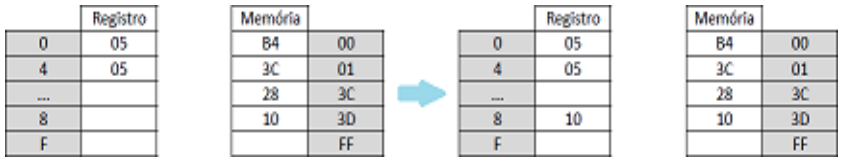


Figura 2.12: Operação B, primeiro caso

Caso o padrão de bits não coincidir:

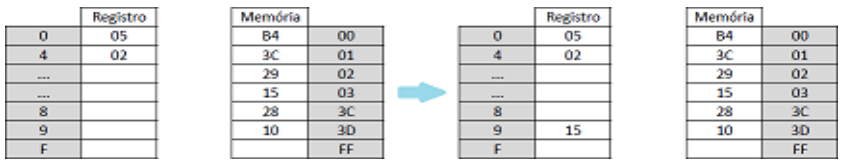


Figura 2.13: Operação B, segundo caso

B43C compara primeiro o conteúdo do registrador 4 com o conteúdo do registrador 0. Se os dois forem iguais, a seqüência de execução será alterada de forma que a próxima instrução a ser executada será aquela localizada no endereço de memória 3C. Caso contrário, a execução do programa continua em sua seqüência normal.

C) HALT (pára) a execução.

Figura 2.14: Operação C

C000 para a execução do programa.

D) PAINT (pinta) o padrão de bits encontrado no registrador R com a cor do conteúdo do registrador S.

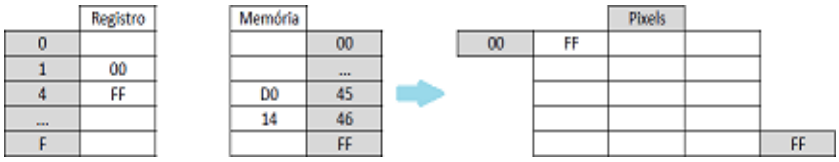


Figura 2.15: Operação D

D014 pinta o conteúdo do registrador 1 com a cor do conteúdo do registrador 4.

2.1.3 Brincando com a máquina

A máquina possui um ciclo de instrução, denominada Ciclo de Von Neumann, na qual a CPU (processador) busca as instruções na memória, decodifica e executa tais instruções. A seguir se encontra um detalhamento de cada etapa desse processo.

A instrução é buscada com base no endereço armazenado no registrador PC (Program Counter). O PC contém o endereço na memória da próxima instrução a ser buscada, decodificada e executada. Quando a instrução é buscada, o endereço armazenado no PC é incrementado para a próxima instrução antes mesmo de ocorrer a decodificação e execução da instrução.

Uma vez acessada a célula ou as células com a instrução, essa é copiada no RI (Registrador de Instrução) para ser decodificada. A decodificação é a etapa em que é determinada qual o tipo da instrução, verificando os 4 primeiros bits para discernir o código de operação da instrução e os dados fornecidos. Algumas instruções não possuem operandos, como a HALT.
Por ultimo, a execução é a fase em que ocorre o procedimento de acordo com o código de operação (em geral utiliza a UC - Unidade de Controle - ou ULA-Unidade Lógica Aritmética). Abaixo se encontra a busca, decodificação e execução da instrução 5314.

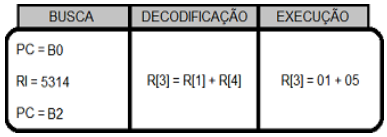


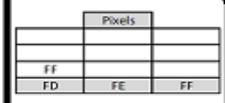
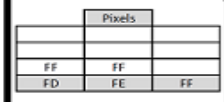
Figura 2.16: Busca, Decodificação e Execução

Na Busca, o PC superior demarca a posição de memória (B0) onde a instrução a ser processada inicia, em seguida a instrução (5314) é armazenada no RI e o PC é incrementado para a posição da próxima instrução (B2). Na Decodificação é definido quem é o operador (5) e quais são os operandos (R[3], R[1] e R[4]). Por fim, na Execução acontece a operação que, neste caso, é soma.

Realizaremos a busca, decodificação e execução da instrução abaixo, considerando que o PC se encontra na posição de memória 12 e o endereçamento do pixel vai de 00 a FF.

Memória	Instruções
10	C0
11	00
12	20
13	FF
14	21
15	FD
16	22
17	FF
18	23
19	00
1A	24
1B	01
1C	25
1D	FF
1E	D0
1F	12
20	51
21	14
22	B1
23	10
24	B0
25	1E

Figura 2.17: Instruções

BUSCA	DECODIFICAÇÃO	EXECUÇÃO
PC = 12 RI = 20FF PC = 14	R[0] = FF	
PC = 14 RI = 21FD PC = 16	R[1] = FD	
PC = 16 RI = 22FF PC = 18	R[2] = FF	
PC = 18 RI = 2300 PC = 1A	R[3] = 00	
PC = 1A RI = 2401 PC = 1C	R[4] = 01	
PC = 1C RI = 25FF PC = 1E	R[5] = FF	
PC = 1E RI = D012 PC = 20	MV[FD] = FF	
PC = 20 RI = 5114 PC = 22	R[1] = R[1] + R[4]	R[1] = FD + 01
PC = 22 RI = B110 PC = 24	SE R[1] = R[0] PC = 10	SE FE = FF NÃO
PC = 24 RI = B01E PC = 26	SE R[0] = R[0] PC = 1E	SE FF = FF SIM
PC = 1E RI = D012 PC = 20	MV[FE] = FF	

BUSCA	DECODIFICAÇÃO	EXECUÇÃO
PC = 20 RI = 5114 PC = 22	R[1] = R[1] + R[4]	R[1] = FE + 01
PC = 22 RI = B110 PC = 24	SE R[1] = R[0] PC = 10	SE FF = FF SIM
PC = 10 RI = C000 PC = 12	HALT	HALT

Figura 2.18: Exemplo do processo de busca, decodificação e execução

2.1.4 Exercícios.

De introdução:

1)Realizar a soma dos conteúdos dos registradores A e B e colocar o resultado no registrador 0.

2)Decrementar 01 do conteúdo do registrador 3, sabendo que R[3] = 05.

3)Multiplicar o conteúdo do registrador 2 pelo do registrador 7, sabendo que R[2] = 05 e R[7] = 04.

4)Realizar o AND binário entre os registradores 3 e 5. O resultado deve ser posto no registrador 4.

5)Realizar o OR binário entre os registradores 8 e A. O resultado deve ser posto no registrador 9.

6)Pintar o pixel do valor do registrador 9 pela cor do valor do registrador 2.

7)Realizar busca, decodificação e execução do conjuntro de instruções a seguir, considerando que PC = 0F:

8)Escrever um conjunto de instruções para a máquina de 16 bits de Glenn, considerando que o valor dos registradores 5 e 4 são aleatórios e entre 00 e 0F. Mostre no registrador 6 o valor do registrador 5 multiplicado pelo do registrador 4.
Use o código de operação B para realizar a multiplicação e finalize o pro-

Memória	Instruções
0E	21
0F	22
A0	02
A1	2F
A2	A0
A3	2B
A4	A2
A5	20
A6	A0
A7	BF
A8	AB
A9	C0
AA	00
AB	50
AC	02
AD	BB
AE	A7
AF	2F
B0	A3
B1	2B
B2	A1
B3	C0
B4	00

grama com C000.

Mais complexos:

1)Escrever um conjunto de instruções para a máquina de 16 bits, considerando que os valores dos registradores 5 e A são aleatórios e maiores que 00:
a) Copiar o valor da soma dos registradores 5 e A para o registrador B;
b) Parar quando o valor do registrador B chegar a FF;
c) Se não chegar a FF, somar o valor 1, a cada iteração, ao registrador 5 e voltar a comparar.

2)Considere a máquina:
a) Os valores dos registradores A e B são desconhecidos e aleatórios;
b) Comparar o valor de A e B, o menor deverá ir para o registrador C e o maior para o D.

3)Imprima a cor branca nos pixels pares e a cor preta nos pixels impares, como num tabuleiro de xadrez, considerando que:
a) A cor branca equivale a 00 e a preta a FF;
b) O primeiro pixel seja 00 e o ultimo FF.

4)Considerando que a instrução D0RS preenche o pixel informado no registrador R com a cor informada no registrador S, e que PC = 12, realizar busca, decodificação com Ri e execucao. O endereçamento do pixel vai de 00 a FF.

Memória	Instruções
10	C0
11	00
12	20
13	0F
14	21
15	00
16	22
17	FF
18	23
19	0D
1A	24
1B	50
1C	25
1D	01
1E	D0
1F	32
20	D0
21	31
22	53
23	35

Memória	Instruções
24	D0
25	32
26	B3
27	2A
28	B0
29	20
2A	20
2B	FF
2C	D0
2D	31
2E	53
2F	34
30	D0
31	32
32	B3
33	10
34	B0
35	2C

2.2 Alan Turing e Von Neumann

A Computação é um campo relativamente novo, do qual muitos debatem sua classificação como ciência. Ainda assim, para chegarmos ao ponto atual, a Computação recebeu a ajuda de inúmeros matemáticos e cientistas, dos quais os mais significativos foram Von Neumann e Alan Turing.

2.2.1 Alan Mathison Turing.

Alan Mathison Turing foi um matemático britânico, nascido em 1912, conhecido como o pai da Computação e pioneiro no campo da Inteligência Artificial. Colaborou para a formação do conceito de algoritmo com a máquina de Turing. Turing trabalhou para a inteligência britânica durante a Segunda Guerra Mundial, onde utilizou de suas teorias para ajudar seu país

durante a guerra. Auxiliou vários campos da ciência, principalmente a computação com a invenção da Máquina de Turing. Alan Turing morreu aos 41 anos em 1954.

- Teste de Turing

É um teste com intuito de examinar a capacidade da inteligência artificial, aspirando conceber uma máquina capaz de reproduzir a inteligência humana. É um experimento em que um usuário realiza perguntas á uma máquina e analisa as respostas, averiguando se elas foram ou não geradas por um ser humano. O sucesso do teste é constatado quando o usuário não consegue fazer a distinção humano-máquina.

2.2.2 Máquina de Turing

Figura 2.19: Máquina de Turing elaborada por Mike Davey, fundador do site aturingmachine.com.

É uma máquina teórica, também conhecida como máquina universal, concebida por Turing. Ela visava somente o funcionamento lógico de um computador, ignorando as partes físicas dele. Foi um conceito elaborado ante a invenção dos computadores.

A máquina continha uma fita, um cabeçote, uma tabela de ação e um registrador de estados para manipular uma quantidade finita de dados dada a quantidade limitada de fita (que seria ilimitada na teoria de Turing). Na máquina, a fita possuia células adjacentes em que o cabeçote se movimentava, lendo, escrevendo e apagando números conforme o algoritmo processado, o registrador de estados trabalhava armazenando o estado da máquina e a tabela de ação consistia dos símbolos utilizados e do meio de movimentação

do cabeçote, além de identificar qual é e qual será o estado da máquina após cada instrução. A seguir se encontra uma tabela dos símbolos usados na tabela de ação de Turing.

Tabela 2.2: Simbologia da Máquina de Turing

Símbolos	Descrição
Q	Conjunto finito de estados
Σ	Conjunto finito de símbolos de entrada
q_0	Estado inicial
q_f	Estado final
R	Right/direita
L	Left/esquerda
A/B/C/etc	Dígitos
β	Vazio/branco
δ	Função de transição
k	Número de fitas

As instruções da máquina de Turing possuem o seguinte formato: função de transição (estado, símbolo) = (estado, símbolo, direção). Abaixo esta um exemplo:

$$\delta(q_0, A) = (q_0, B, R)$$

Nela, a instrução informa que quando o cabeçote ler o digito 'A' no estado inicial, ele ira manter seu estado, reescrever 'B' em seu lugar e se mover para a direita.

Testando a Máquina

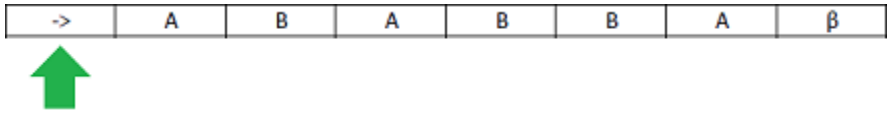
Na máquina abaixo, vamos ver se ela reconhece se a quantidade de A's é igual a quantidade de B's.

Instruções:

$$\delta(q_{\rightarrow}, \rightarrow) = (q_0, \rightarrow, R) \qquad \delta(q_1, A) = (q_x, X, L) \qquad \delta(q_2, X) = (q_2, X, R)$$
$$\delta(q_0, A) = (q_2, X, R) \qquad \delta(q_1, B) = (q_1, B, R) \qquad \delta(q_x, A) = (q_x, A, L)$$
$$\delta(q_0, B) = (q_1, X, R) \qquad \delta(q_1, X) = (q_1, X, R) \qquad \delta(q_x, B) = (q_x, B, L)$$
$$\delta(q_0, X) = (q_0, X, R) \qquad \delta(q_2, A) = (q_2, A, R) \qquad \delta(q_x, X) = (q_0, X, R)$$

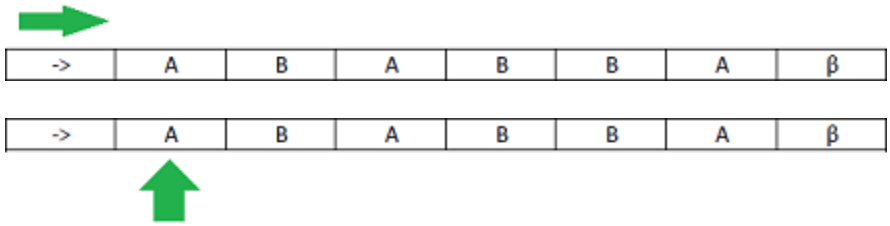
$$\delta(q_0, \beta) = (q_f, \beta, R) \qquad \delta(q_2, B) = (q_x, X, L)$$

Vamos considerar que a máquina começa a ler no símbolo $\rightarrow$.



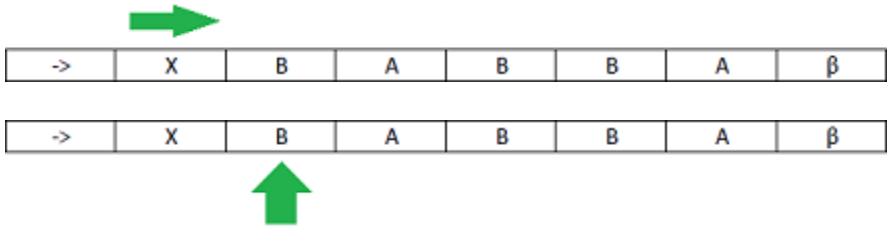
$$\delta(q_{\rightarrow}, \rightarrow) = (q_0, \rightarrow, R)$$

No estado de transição $q_{\rightarrow}$, ao ler $\rightarrow$, vai para q_0, reescreve $\rightarrow$ em seu lugar e se movimenta para a direita.



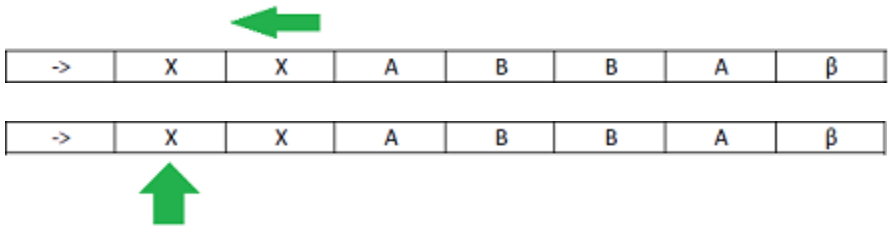
$$\delta(q_0, A) = (q_2, X, R)$$

No estado de transição q_0, ao ler A, vai para q_2, reescreve X em seu lugar e se movimenta para a direita.



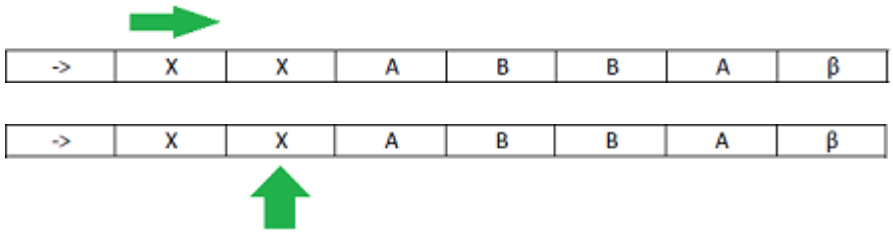
$$\delta(q_2, B) = (q_x, X, L)$$

No estado de transição q_2, ao ler B, vai para q_x, reescreve X em seu lugar e se movimenta para a esquerda.



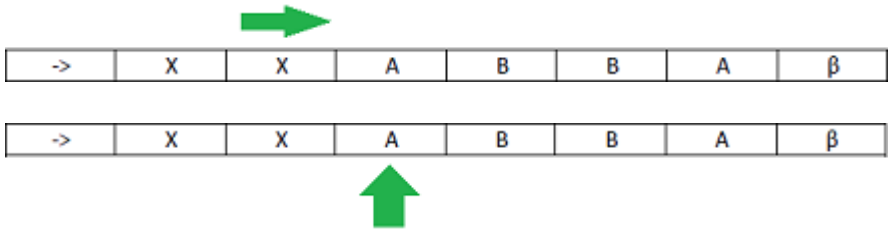
$$\delta(q_x, X) = (q_0, X, R)$$

No estado de transição q_x, ao ler X, vai para q_0, reescreve X em seu lugar e se movimenta para a direita.



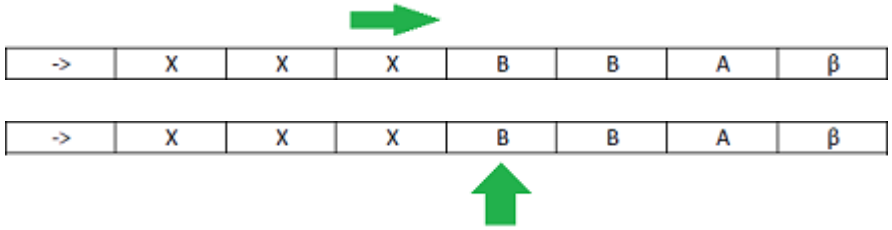
$$\delta(q_0, X) = (q_0, X, R)$$

No estado de transição q_0, ao ler X, se mantém em q_0, reescreve X em seu lugar e se movimenta para a direita.



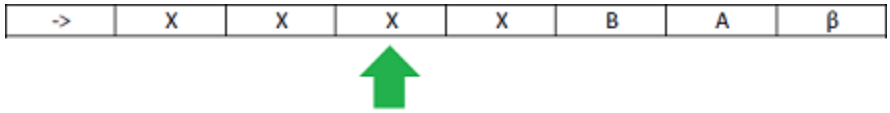
$$\delta(q_0, A) = (q_2, X, R)$$

No estado de transição q_0, ao ler A, vai para q_2, reescreve X em seu lugar e se movimenta para a direita.



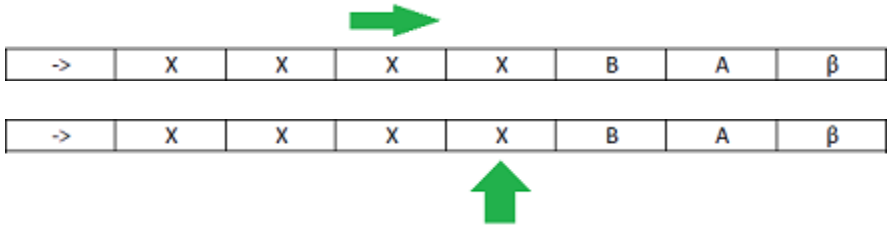
$$\delta(q_2, B) = (q_x, X, L)$$

No estado de transição q_2, ao ler B, vai para q_x, reescreve X em seu lugar e se movimenta para a esquerda.



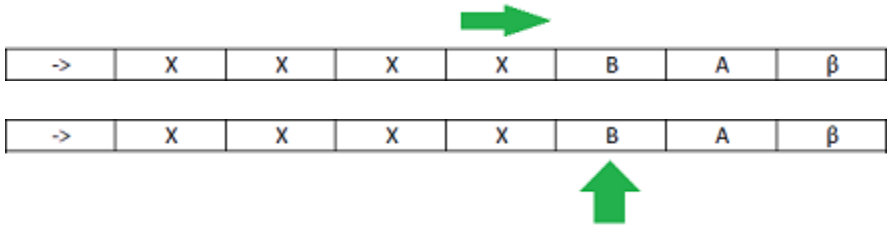
$$\delta(q_x, X) = (q_0, X, R)$$

No estado de transição q_x, ao ler X, vai para q_0, reescreve X em seu lugar e se movimenta para a direita.



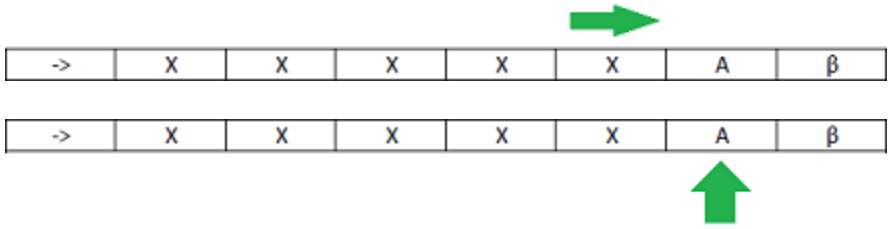
$$\delta(q_0, X) = (q_0, X, R)$$

No estado de transição q_0, ao ler X, se mantém em q_0, reescreve X em seu lugar e se movimenta para a direita.



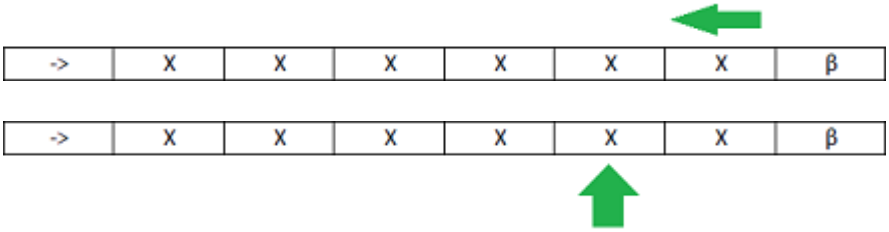
$$\delta(q_0, B) = (q_1, X, R)$$

No estado de transição q_0, ao ler B, vai para q_1, reescreve X em seu lugar e se movimenta para a direita.



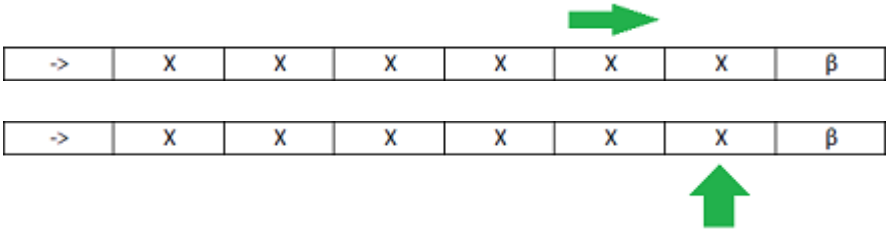
$$\delta(q_1, A) = (q_x, X, L)$$

No estado de transição q_1, ao ler A, vai para q_x,reescreve X em seu lugar e se movimenta para a esquerda.



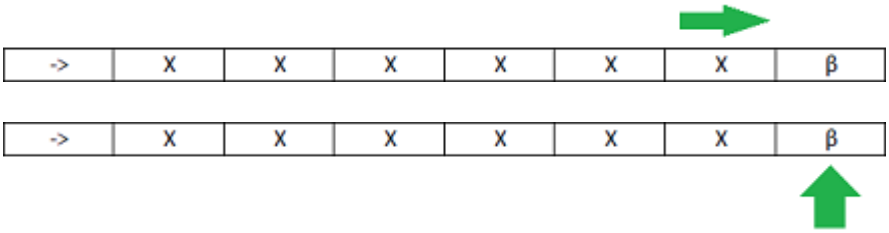
$$\delta(q_x, X) = (q_0, X, R)$$

No estado de transição q_x, ao ler X, vai para q_0,reescreve X em seu lugar e se movimenta para a direita.



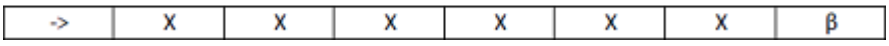
$$\delta(q_0, X) = (q_0, X, R)$$

No estado de transição q_0, ao ler X, se mantém em q_0,reescreve X em seu lugar e se movimenta para a direita.



$$\delta(q_0, \beta) = (q_f, \beta, R)$$

No estado de transição q_0, ao ler β, vai para q_f, reescreve β em seu lugar e finaliza o processo.



Podemos concluir que a máquina reconheceu que a quantidade de A's foi equivalente a de B's.

2.2.3 John Von Neumann

John Von Neumann foi um matemático húngaro nascido em 1903, e posteriormente naturalizado nos Estados Unidos, que trouxe diversas contribuições para a evolução da Computação, das quais algumas das mais importantes foram a criação da ordem de processamento, baseada em números binários em vez de decimais, que é utilizada até a atualidade, o Ciclo de Von Neumann (Busca, Decodificação e Execução) e a arquitetura de Von Neumann, o modelo em que os computadores atuais são projetados, embora sofram de pequenas modificações.

Arquitetura de Von Neumann

Nomeada Arquitetura de Von Neumann, a arquitetura de computador é composta por 4 componentes principais, eles são:

- A memória: local onde são armazenados os dados (em formato binário);
- A unidade aritmética e lógica (ULA): tem como função a execução de cálculos aritméticos e booleanos. É apoiado por registradores;
- A unidade de controle (UC): tem como função buscar instruções na memória principal e identificar qual o tipo delas;
- A entrada e saída: são dispositivos físicos que entram ou recebem informações no computador. Exemplo: teclado, mouse, webcam, entre outros.

A imagem a seguir ilustra a Arquitetura de Von Neumann.

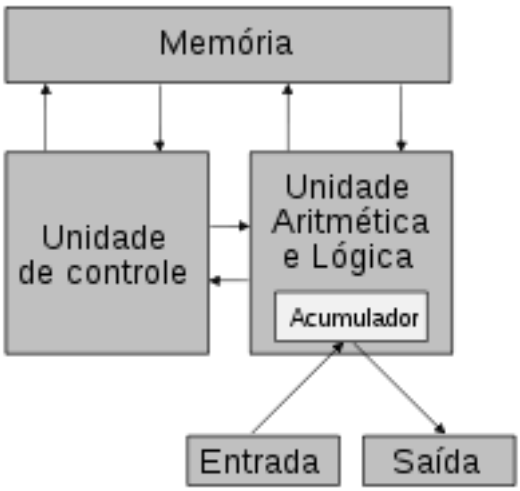


Figura 2.20: Arquitetura de Von Neumann, retirada da página Wikimedia Commons.

Na proposta original existia uma ULA e UC que agora estão na CPU.

Uma das qualidades desse modelo é a utilização do programa armazenado, que é a capacidade de armazenar programas no mesmo espaço de memória que os dados, um conceito somente teórico ante a invenção da arquitetura de Von Neumann.

O armazenamento das instruções na memória foi um enorme avanço que tornou o computador mais rápido e versátil devido a fácil obtenção de instruções que, na época, era um processo árduo. Além disso, essa característica permite a modificação de programas durante sua execução. Ainda assim, a arquitetura de programa armazenado possui desvantagens, sendo eles os erros que podem ocorrer devido a programas defeituosos, que podem acarretar problemas em outros programas ou até mesmo colapso do sistema operacional, e o Gargalo de Von Neumann.

O Gargalo de Von Neumann se refere ao limite de transferência de dados entre a memória e o CPU, que é inferior a quantidade que o processador é capaz de processar. Essa diferença de dados força o CPU a esperar a recuperação de dados da memória para que ocorra o processamento, causando perda de tempo.

2.2.4 Outras contribuições

- Joseph Marie Jacquard

Apesar de Turing ser considerado o pai da Computação, o primeiro compu-

tador a ser efetivamente construído foi o tear Jacquard, nomeado após seu inventor, o mecânico Joseph Marie Jacquard, nascido em 1752 na França. Diferente dos vários contribuidores da computação, Jacquard não foi um cientista, mas sim um tecelão que buscou criar um meio de evitar as inconveniências acarretadas pelo trabalho manual do tear.

O tear automático de Jacquard foi uma máquina têxtil, concebida em 1804, que substituiu a necessidade de trabalho humano pelo autônomo. O dispositivo utilizava de cartões perfurados para produzir padrões têxteis complexos e sem falhas. O mecanismo de Jacquard pode ser encontrado no Museu de Artes e Ofícios - Musée des arts et métiers - em Paris. Anos mais tarde, Charles Babbage projetou sua Máquina Diferencial, onde utilizaria de cartões perfurados, seguindo a idéia de Jacquard.

- Charles Babbage

Outro fato interessante e pouco conhecido entre os fundadores do computador moderno é que o primeiro projeto de computador de uso geral, denominado de Máquina Diferencial, foi inventado por Charles Babbage, um cientista inglês nascido em 1791, que sentia a necessidade de eliminar as falhas humanas presentes em cálculos.

Babbage, junto de Ada Lovelace, conhecida como Condessa de Lovelace, a colaboradora no desenvolvimento de algoritmos voltados para a máquina, planejavam conceber um aparelho que, ao receber dados, realizaria cálculos de funções polinomiais controlados por cartões. O projeto de Babbage, infelizmente, era muito avançado para a época e o cientista não possuía verba o suficiente para produzir sua invenção, todavia, tanto Babbage quanto Ada receberam recognição pela influência que o projeto trouxe, sendo reconhecidos como o pai do computador e a primeira programadora da história, respectivamente.

Além disso, a sua máquina analítica, apresentada em 1833, é considerada a primeira referencia para os computadores modernos eletrônicos.

Figura 2.21: Máquina diferencial de Babbage, construida pelo Science Museum, em Londres

3

Capítulo 3 - Memória

3.1 Pirâmide de memória

Memória, na informática, são componentes do computador que têm como função armazenar dados e informações para que elas sejam usadas quando necessárias. Alguns tipos de memória, como a principal, são indispensáveis para o funcionamento do sistema. A unidade de medida usada na memória é a binária, da qual todos os modelos de arquitetura, como o de Von Neumann, seguem.

Os principais tipos de memória são a memória principal (RAM), a memória secundária, a memória cache e os registradores, todos presentes na imagem abaixo:

Figura 3.1: Hierarquia de memória, imagem modificada do site howstuffworks.com

Hierarquia de memória é um conceito concebido para se referir a relação entre os diferentes tipos de memória e suas características, essa geralmente

aludindo as principais que são o custo, a capacidade de armazenamento e o tempo de acesso. Em outros casos ela também faz menção a volatilidade, tecnologia e temporariedade da informação.

É interessante notar que há uma diferença entre Memória e Armazenamento, a primeira se refere a componentes que perdem seu conteúdo perante o desligamento do sistema, enquando o segundo se refere a dispositivos que não perdem.

Neste capítulo abordaremos os principais tipos de memória e seus aspectos presentes num sistema computacional.

3.2 Memória interna

3.2.1 Registradores

Registradores são pequenas porções de memória temporárias localizadas no CPU que permitem aumentar a velocidade de execução dos programas. Alguns registradores tem funções específicas, como o PC e o RI, ambos previamente introduzidos. Por serem localizados no topo da pirâmidade, eles tem uma baixa capacidade de armazenamento, mas possuem a mais rápida velocidade de acesso aos dados se comparada a outros tipos de memória.

3.2.2 Memória Principal

A memória principal, a memória RAM como previamente discutida, ou até Memória de Leitura e Escrita, é a memória vital para o funcionamento do sistema computacional. Ela tem papel de armazenar todas as informações que o processador necessitar em qualquer momento, por isso seu nome. Esse tipo de memória tem acesso direto ao processador, servindo como uma ponte para a memória secundária poder ser endereçada pelo processador.

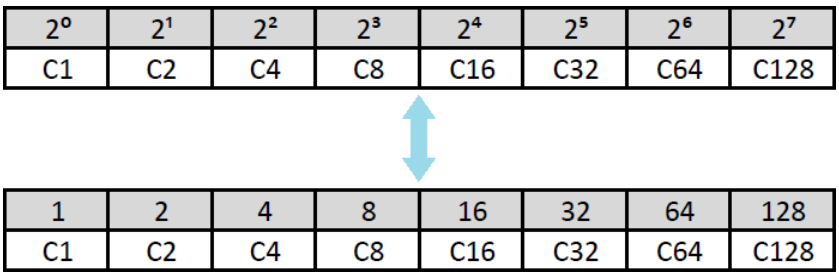
Dentro da memória principal, um dos temas mais importantes se remete a detecção e correção de erros de dados, de onde foram elaborados diversos metódos para realizarem essas tarefas (metódo de paridade, metódo de repetição, checksum, entre outros), dos quais muitos não obtém êxito. Na próxima seção abordaremos um dos metódos mais eficazes na área: o Algoritmo Hamming.

3.2.3 Correção de Erro - Algoritmo Hamming

Desenvolvido por Richard Wesley Hamming, o algoritmo Hamming, denominado de algoritmo de correção, realiza a detecção e correção de erros (conhecido como ECC: “Error Correcting Code") na transferência e armazenamento de dados. Na transferência, os dados enviados são perdidos quando chegam a seu destino, o código de Hamming é uma forma simples de detectar se houve erro no deslocamento em até dois bits e, se houver, corrige um único bit.

Um dos maiores problemas enfrentados pelos computadores atuais são os erros de memória que podem interromper os processos do sistema, fazendo com que o usuário, para emendar os problemas causados pelo erro, tenha que reiniciar sua máquina, ás vezes perdendo progresso em suas atividades. Outra adversidade consequente dos erros de memória é a vulnerabilidade nos sistemas de segurança, que são ameaçados por bugs. A eficiência do código de Hamming em corrigir e evitar essas inconveniências o torna o corretor de erros mais usado atualmente.

O algoritmo usa de bits de paridade para averiguar se houve êxito ou não na transferência de dados, esses bits são intitulados de c1, c2, c4, c8, em diante. Lembrando de como transformar decimal para binário podemos associar os bits de paridade transformando-os em binários. Isto é, c1 equivale a 2^0 que é igual 1, c2 equivale a 2^1 que é igual a 2, c4 equivale a 2^2 que é igual a 4, c8 equivale a 2^3 que é igual a 8, e assim sucessivamente. A imagem abaixo ilustra essa associação.



No código Hamming, também dispomos das posições de memórias que são os dados transferidos a ser verificados pelo algoritmo, elas são nomeadas de M1, M2, M3, M4 e assim consecutivamente.

Abaixo segue uma breve explicação de como encontrar os bits de paridade (c1, c2, c4, c8, etc.), fazendo uso de índices (1, 2, 3, 4, e assim em diante) para auxiliar na explicação .

C1	C2	M1	C4	M2	M3	M4	C8	M5	M6	M7	M8	M9	M10	M11
-	-	1	-	0	1	1	-	0	1	1	0	1	0	0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

M1 está no índice 3, para representar 3 decimal em binário temos 0000 0011, logo M1 pertence a c1 e c2. M4 está no índice 7, para representar 7 decimal em binário temos 0000 0111, logo M7 pertence a c1, c2 e c4. M11 está no índice 15, para representar 15 decimal em binário temos 0000 1111, logo M15 pertence a c1,c2,c4 e c8. Lembrando que só podemos associar aos índices as posições de memória (M1, M2, M3, M4, M5, etc...).

Após descobrir quais posições de memória pertencem a cada bit, realizaremos o $\oplus$ (OU EXCLUSIVO) nos bits relacionados a cada posição para descobrir os valores dos bits de paridade. Por exemplo, c4 tem M2, M3, M4, M8, M9, M10, M11 relacionados a si, então ao realizar o $\oplus$ nestes bits nós descobriremos o valor de c4. Aplica-se o mesmo processo para descobrir o valor de qualquer bit de paridade.
c1 = 1 0 1 0 1 1 0 = 1 1 0 1 1 0 = 0 0 1 1 0 = 0 1 1 0 = 1 1 0 = 0 0 = 0
c2 = 1 1 1 1 1 0 0 = 0 1 1 1 0 0 = 1 1 1 0 0 = 0 1 0 0 = 1 0 0 = 1 0 = 1
c4 = 0 1 1 0 1 0 0 = 1 1 0 1 0 0 = 0 0 1 0 0 = 0 1 0 0 = 1 0 0 = 1 0 = 1
c8 = 0 1 1 0 1 0 0 = 1 1 0 1 0 0 = 0 0 1 0 0 = 0 1 0 0 = 1 0 0 = 1 0 = 1

C1	C2	C4	C8
0	1	1	1
1	2	4	8

Podemos concluir que:

$$c1 = M1 \oplus M2 \oplus M4 \oplus M5 \oplus M7 \oplus ...$$

$$c2 = M1 \oplus M3 \oplus M4 \oplus M6 \oplus M7 \oplus ...$$

$$c4 = M2 \oplus M3 \oplus M4 \oplus M8 \oplus M9 \oplus ...$$

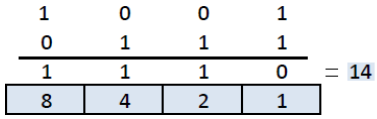
$$c8 = M5 \oplus M6 \oplus M47 \oplus M8 \oplus M9 \oplus ...$$

Em diante.

Um fato interessante que podemos tirar dessa conclusão é que a c1 estão relacionados todos os índices ímpares. Por exemplo M1, M2 e M4 ocupam os índices 3, 5 e 7, respectivamente, o que os tornam associados ao bit c1. Também realizamos que não é necessário calcular as posições de memória que possuam valor 0, poís as mesmas não influenciam a operação.

O erro na transferência é detectado quando, após calcularmos os $\oplus$ das posições de memória, um ou mais bits de paridade estão diferentes do valor inicial. Para verificar exatamente onde houve o erro basta fazermos o $\oplus$ dos bits de paridade iniciais e dos corrigidos e somar as posições iguais a 1.

Caso os bits de paridade recebidos no exemplo anterior fossem c1 = 1, c2 = 0, c4 = 0 e c8 = 1, o $\oplus$ dos bits recebidos e dos calculados ficaria como mostra a imagem:



O erro foi identificado no índice 14 que corresponde a posição de memória M10. Como a unidade de medida utilizada é a binária, para corrigir o erro basta trocar a unidade de M10 que neste caso é 0 para 1.

Exercícios:

1) Utilizando o código de Hamming, complete a seguinte sequência de bits considerando que o bit de correção C1 esta a esquerda.

		1		1	1	0		1	1	0	0	1	0	1

2) Faça a correção da sequência de bits a seguir considerando que o bit de correção C1 esta a direita.

0	0	1	1	1	1	1	0	0	1	0	1	1	0	0

3.3 Memória Cache

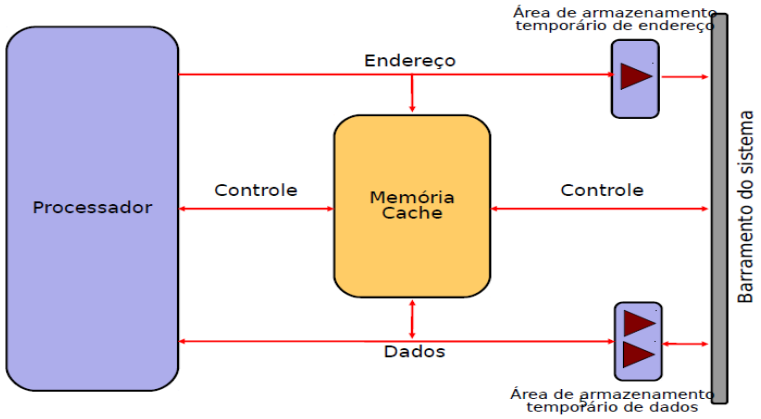


Figura 3.2: Organização da memória cache, imagem modificada do livro de William Stallings

Memória cache é uma memória temporária que auxilia na transferência de dados entre o CPU e a memória principal. Devido ao limite de dados que a memória envia para o CPU, como mencionado no Gargalo de Von Neumann, o processador passa períodos sem realizar trabalho criando um gargalo, a memória cache foi a solução elaborada para esse problema, ela fornece ao processador os dados para que ele evite buscar várias vezes na memória RAM, assim melhorando o desempenho do sistema.

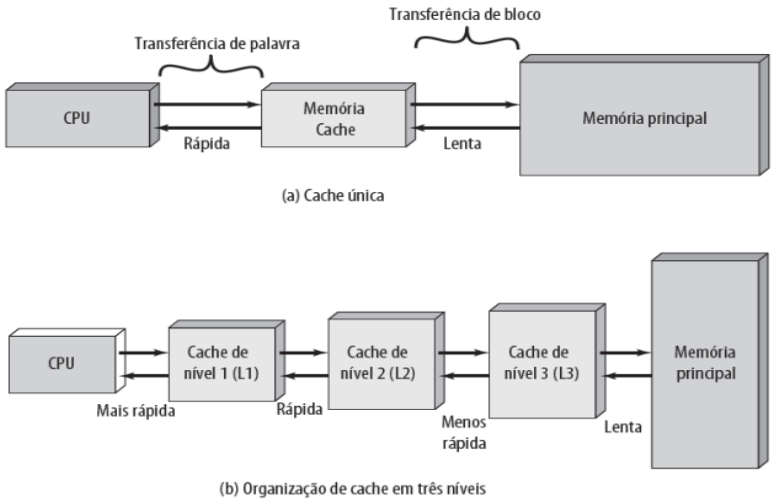


Figura 3.3: Organização da cache única e com níveis, imagem retirada do livro de William Stallings

A memória cache possui maior velocidade de transferência que a memória RAM por ser um intermédio dela e do processador. A cache armazena informações frequentemente acessadas para que o processador as possa usar futuramente. Ela também é opcional, em alguns computadores ainda não é encontrado memória cache, e difere a cada modelo, sendo encontrada em diversos dispositivos (processadores, servidores, placas-mãe, etc...). Devido a seu alto custo de produção, a cache foi dividida até 3 níveis: L1, L2 e L3, em que as duas primeiras são internas ao processador enquanto a ultima é externa, junta a placa-mãe.

Como a estrutura da memória cache é dependente de seu fabricante, em alguns casos ela é grande, fazendo com que seu tempo de acesso seja próximo a da memória total, e, em outros, pequena, buscando coincidir seu custo por bits com o da memória principal. Algumas vantagens da minimização do tamanho da cachê é que, quanto menor, melhor será seu endereçamento devido a um menor número de pinos. Também é necessário ter em mente o espaço limitado na placa de circuitos.

A memória principal é composta de blocos que possuem N palavras, onde cada palavra possui um endereço distinto. A memória cachê é composta de blocos que consistem em linhas da qual cada uma possui N palavras mais um TAG (endereço da palavra) de alguns bits.

Conversão de dados

Essa seção irá abordar a conversão entre as unidades de informação (bits, bytes, kilo, mega...) para dispor uma base para o aluno realizar cálculos futuros deste capítulo. Abaixo se encontra uma tabela para auxílio na conversão.

Conversão de dados	
Tipo de dados (1 unidade)	Tamanho
Terabyte	1.000.000.000.000 bytes
Gigabyte	1.000.000.000 bytes
Megabyte	1.000.000 bytes
Kilobyte	1.000 bytes
Byte	8 bits
Bit	binário 1 ou 0

Por exemplo, 256 bytes corresponde a 2^8, 256 Kilobytes a 2^{18}, 256 megabytes a 2^{28} e 256 gigabytes a 2^{38}. Com isso podemos concluir que $1KB = 2^{10}$, $1MB = 2^{20}$ e $1GB = 2^{30}$. Também é possível deduzir que, como 1 byte equivale a 8 bits, isto é, 2^0 bytes é igual a 2^3 bits, a casa da potência se move 3 unidades à transição: 2^{25} bytes corresponde a 2^{28} bits e 2^{16} bits corresponde a 2^{13} bytes.

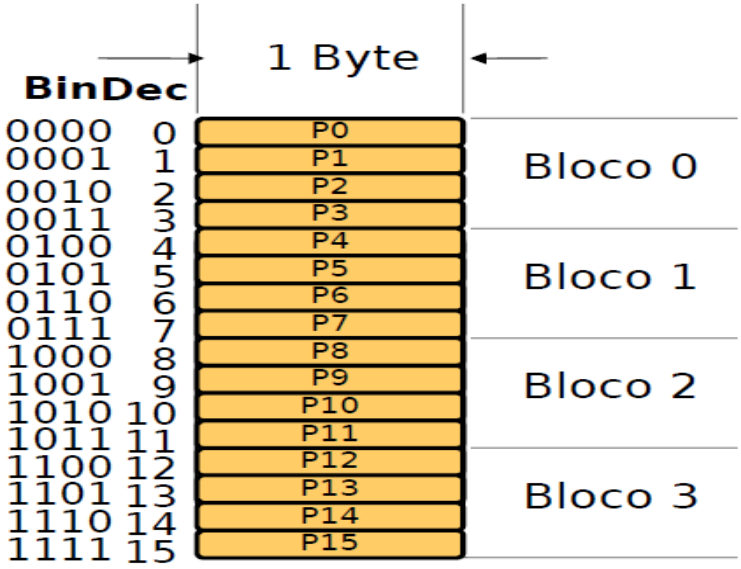
3.3.1 Mapeamento

Mapeamento é o processo em que identificamos os elementos de um endereço na memória cachê ou na principal, verificando se ele existe ou não em um bloco. Esse processo se vê necessário devido ao número de linhas da cache ser menor que a quantidade de blocos na principal. Existem três tipos de técnicas de mapeamento na memória cachê, elas são: mapeamento direto, mapeamento associativo e mapeamento associativo por conjunto.

Mapeamente direto

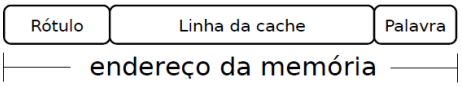
O mapeamento direto é um tipo de mapeamento simples de baixo custo de implementação. Uma desvantagem de seu uso é que, quando o programa referênciar a mesma palavra em blocos distintos, mas em linhas iguais, várias vezes, os blocos serão alterados repetidamente e a taxa de acerto na cache terá sua precisão drasticamente reduzida.

Nele existem a Tag (o rótulo), a coluna e a linha, em que a quantidade de linhas do bloco da memória RAM equivale a quantidade de células do bloco/linha da memória Cache.



Por exemplo, a imagem acima ilustra um computador com uma memória principal de 16 bytes, onde cada byte é endereçável diretamente. A memória principal pode ser vista como 4 blocos de 4 bytes cada e a memória cache, de 8 bytes, é organizada em 2 linhas, cada qual com 4 bytes.

De acordo com as imagens e informações anteriores, podemos descobrir como identificar os elementos do endereço pelas formas a seguir:



- Coluna: divisão do tamanho da linha pelo tamanho de cada célula;
- Linha: divisão do tamanho do cache pelo tamanho de cada linha;
- Tag (Rótulo): divisão do tamanho da memória principal pelo tamanho

do cache ou tamanho do endereço menos o somatório do tamanho da linha e da coluna;

- Endereço: divisão da memória principal pelo tamanho de cada célula.

Também podemos descobrir a qual bloco pertence i linha do cache pela fórmula $i = j\%m$, em que i é igual a linha, j é igual ao bloco e m corresponde ao número de linhas.

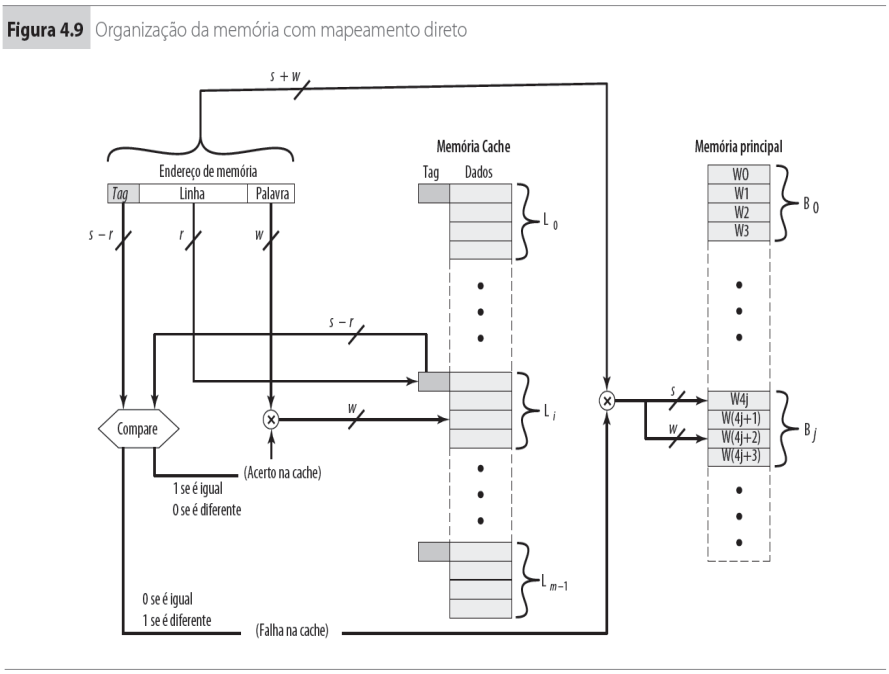


Figura 3.4: Funcionamento do mapeamento direto, imagem retirada do livro de Arquitetura e Organização de Computadores de William Stallings. Nela, s e r correspondem á tags, r á linha e w á palavra dentro da linha do cache (coluna)

Exemplo:

Um computador possui um memória principal com capacidade para 2 Gigabytes. Cada célula desta memória tem capacidade para 1 byte. Foi colocada neste computador uma memória cache de mapeamento direto com capacidade para 512 kbytes. Cada linha desta cache tem capacidade para 16 células. Supondo que a CPU faça um acesso ao endereço $(035AFBE5)_{16}$, calcule:
a) Total de bits do endereço;
b) Total de bits para o número da coluna;
c) Total de bits para o número da linha;

d) Total de bits para a TAG;
e) O número da coluna (em hexadecimal);
f) O número da linha (em hexadecimal);
g) O valor da Tag (em hexadecimal).

Sabemos que 2 Gigabytes correspondem a 2048 Mbytes que é igual a 2^{28} bytes e que 512 kbytes é igual a 2^{19} bytes, também podemos inferir que cada linha do cache possui 16 bytes, poís cada linha possui 16 células com cada uma medindo 1 byte. Agora calcularemos o tamanho do endereço:

$$Endereço = 2^{28}/2^0 \rightarrow Endereço = 2^{28} = 28$$

Em seguida o total de bits para o número da coluna:

$$Coluna = 2^4/2^0 \rightarrow Coluna = 2^4 = 4$$

O total de bits para o número da linha:

$$Linha = 2^{19}/2^4 \rightarrow Linha = 2^{15} = 15$$

E por ultimo, o total de bits para a TAG:

$$Tag = 2^{28}/2^{19} \rightarrow Tag = 2^9 = 9$$

Para acharmos a coluna, linha e Tag do endereço $(035AFBE5)_{16}$ começaremos transformando o endereço para binário, onde ele ficara que nem abaixo:

0	3	5	A	F	B	E	5
0000	0011	0101	1010	1111	1011	1110	0101

Como sabemos que nessa memória cache, a TAG, linha e coluna ocupam os espaços de 9 bits, 15 bits e 4 bits, respectivamente, podemos separar o endereço que nem o seguinte:

TAG	Linha	Coluna
001101011	010111110111110	0101

=

06B	2FBE	5

Onde a TAG corresponde a 06B, a linha a 2FBE e a coluna a 5.

Exercícios:

1) Um computador possui uma memória principal com capacidade para 4 Gbytes. Cada célula desta memória tem capacidade para 2 bytes. Foi colocada neste computador uma memória cache de mapeamento direto com

capacidade para 128 Mbits. Cada linha desta cache tem capacidade para 64 células. Supondo que a CPU faça um acesso ao endereço $(09FF\ DB0F)_{16}$, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para o número da linha;
d) O total de bits para a Tag;
e) O número da coluna (em hexadecimal);
f) O número da linha (em hexadecimal);
g) O valor da Tag (em hexadecimal).

2) Um computador possui uma memória principal com capacidade para 2 Gbits. Cada célula desta memória tem capacidade para 2 bytes. Foi colocada neste computador uma memória cache de mapeamento direto com capacidade para 1 Mbyte. Cada linha desta cache tem capacidade para 512 bits. Supondo que a CPU faça um acesso ao endereço $(06EC\ 78AE)_{16}$, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para o número da linha;
d) O total de bits para a Tag;
e) O número da coluna (em hexadecimal);
f) O número da linha (em hexadecimal);
g) O valor da Tag (em hexadecimal).

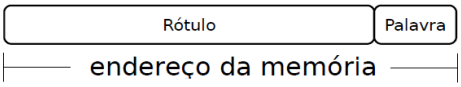
3) Um computador possui uma memória principal com capacidade para 2 Gbytes. O Barramento de Endereços deste computador possui 31 bits. Foi colocado nele uma memória cache de mapeamento direto com capacidade para 1 Mbytes. Cada linha desta cache tem capacidade para 512 bits. Supondo que a CPU faça um acesso ao endereço $(037D\ 6BC5)_{16}$, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para o número da linha;
d) O total de bits para a Tag;
e) O número da coluna (em hexadecimal);
f) O número da linha (em hexadecimal);
g) O valor da Tag (em hexadecimal).

Mapeamento associativo

O mapeamento associativo proporciona maior flexibilidade na apuração do bloco a ser substituído quando ocorre transferência de blocos entre a memória principal e a cache. Em contrapartida, esse mapeamento possui um alto

nível de complexidade do conjunto de circuitos essenciais para a comparação simultânea das TAGs de todas as linhas da memória cache.

No mapeamente puramente associativo os blocos são divididos somente em TAG e coluna (palavra), com os bits mais significativos do endereço se encontrando na TAG. A localização dos blocos na cache é feita a partir de uma verificação que determina se o bloco existe ou não comparando as TAGs de cada linha. Abaixo se encontra como descobrir o total de bits do(a):



- Coluna (Palavra): divisão do tamanho de cada linha pelo tamanho de cada célula;
- Tag (Rótulo): divisão do tamanho da memória principal pelo tamanho de cada linha;
- Endereço: divisão da memória principal pelo tamanho de cada célula.

Figura 4.11 Organização da memória cache totalmente associativa

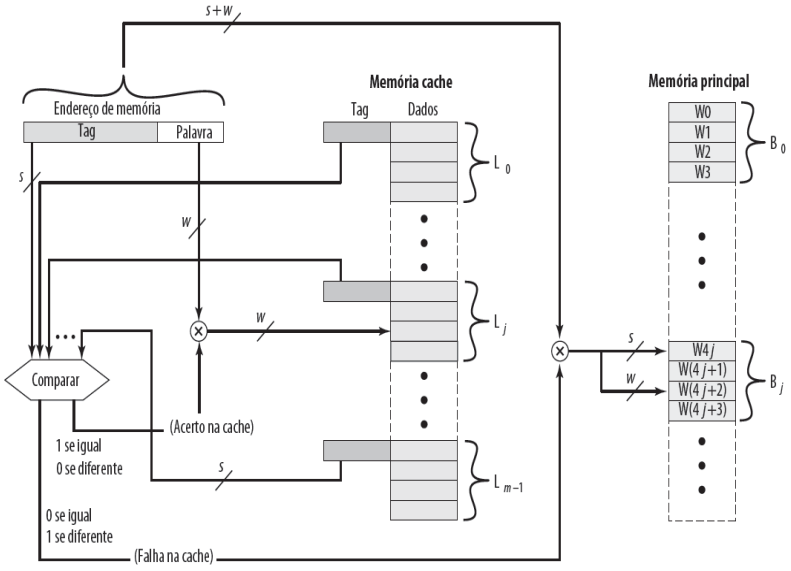


Figura 3.5: Funcionamento do mapeamento associativo, imagem retirada do livro de Arquitetura e Organização de Computadores de William Stallings.

Exemplo:

Um computador possui uma memória principal com capacidade para 4 Gbytes. Cada célula desta memória tem capacidade para 16 bits. Foi colocada neste computador uma memória cache puramente associativa com capacidade para 512 Kbytes. Cada linha desta cache tem capacidade para 64 bytes. Supondo que a CPU faça um acesso ao endereço $(7B7C\ 45DF)_{16}$, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para a Tag;
d) O número da coluna (em hexadecimal);
e) O valor da Tag (em hexadecimal).

Sabemos que 4 Gigabytes correspondem a 2^{32} bytes e que 512 kbytes é igual a 2^{19} bytes. Agora calcularemos o tamanho do endereço:

$$Endereço = 2^{32}/2^1 \rightarrow Endereço = 2^{31} = 31$$

Em seguida o total de bits para o número da coluna/palavra:

$$Coluna = 2^6/2^1 \rightarrow Coluna = 2^5 = 5$$

E por ultimo, o total de bits para a TAG, que podemos achar por:

$$Tag = 2^{31}/2^5 \rightarrow Tag = 2^{26} = 26$$

Para acharmos a coluna, linha e Tag do endereço $(7B7C\ 45DF)_{16}$ começaremos transformando o endereço para binário, onde ele ficara que nem abaixo:

7	B	7	C	4	5	D	F
0111	1011	0111	1100	0100	0101	1101	1111

Como sabemos que nessa memória cache, a TAG e coluna ocupam os espaços de 26 e 5 bits, respectivamente, podemos separar o endereço que nem o seguinte:

TAG	Coluna
01111011011111000100010 1110	11111

=

3DBE22E	1F

Onde a TAG corresponde a 3DBE22E e a coluna a 1F.

Exercícios:

1) Um computador possui uma memória principal com capacidade para 4 Gbytes. Cada célula desta memória tem capacidade para 16 bits. Foi colocada neste computador uma memória cache puramente associativa com capacidade para 512 Kbytes. Cada linha desta cache tem capacidade para 64 bytes. Supondo que a CPU faça um acesso ao endereço $(7B7C\ 45DF)_{16}$, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para a Tag;
d) O número da coluna (em hexadecimal);
e) O valor da Tag (em hexadecimal).

2) Um computador possui uma memória principal com capacidade para 2 Gbits. Cada célula desta memória tem capacidade para 1 byte. Foi colocada neste computador uma memória cache puramente associativa com capacidade para 512 Kbytes. Cada linha desta cache tem capacidade para 16 células. Supondo que a CPU faça um acesso ao endereço $(036D\ 7BC5)_{16}$, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para a Tag;
d) O número da coluna (em hexadecimal);
e) O valor da Tag (em hexadecimal).

3) Um computador possui uma memória principal com capacidade para 2 Gbits. Cada célula desta memória tem capacidade para 2 bytes. Foi colocada neste computador uma memória cache puramente associativa com capacidade para 1 Mbytes. Cada linha desta cache tem capacidade para 512 bits. Supondo que a CPU faça um acesso ao endereço $(07EC\ 98D3)_{16}$, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para a Tag;
d) O número da coluna (em hexadecimal);
e) O valor da Tag (em hexadecimal).

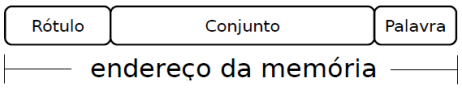
4) Um computador possui uma memória principal com capacidade para 8 Gbits. Cada célula desta memória tem capacidade para 2 bytes. Foi colocada neste computador uma memória cache puramente associativa com capacidade para 1 Mbyte. Cada linha desta cache tem capacidade para 16 células. Supondo que a CPU faça um acesso ao endereço $(195F\ CB7E)_{16}$, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;

c) O total de bits para a Tag;
d) O número da coluna (em hexadecimal);
e) O valor da Tag (em hexadecimal).

5) Um computador possui uma memória principal com capacidade para 16 Gbits. O Barramento de Endereços deste computador possui 30 bits. Foi colocado nele uma memória cache puramente associativa com capacidade para 1 Mbytes. Cada linha desta cache tem capacidade para 512 bits. Supondo que a CPU faça um acesso ao endereço $(0359\ 4BD5)_{16}$, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para a Tag;
d) O número da coluna (em hexadecimal);
e) O valor da Tag (em hexadecimal).

6) Um computador possui uma memória principal com capacidade para 4 Gbytes. Cada célula desta memória tem capacidade para 16 bits. Foi colocada neste computador uma memória cache puramente associativa com capacidade para 2 Mbytes. Cada linha desta cache tem capacidade para 64 bytes. Supondo que a CPU faça um acesso ao endereço (1A73C4DE)16, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para a Tag;
d) Qual o número de bits mínimo do barramento de endereço
d) O número da coluna (em hexadecimal);
e) O valor da Tag (em hexadecimal).

Mapeamente associativo por conjunto



Exercícios:

1) Um computador possui uma memória principal com capacidade para 4 Gbytes. Cada célula desta memória tem capacidade para 16 bits. Foi colocada neste computador uma memória cache associativa por conjunto com capacidade para 512 Kbytes. Cada linha desta cache tem capacidade para 64 bytes. Cada conjunto possui 2 linhas. Supondo que a CPU faça um acesso ao endereço $(73A1\ 49DE)_{16}$, calcule:
a) O total de bits do endereço;

b) O total de bits para o número da coluna;
c) O total de bits para o número do conjunto;
d) O total de bits para a Tag;
e) O número da coluna (em hexadecimal);
f) O número do conjunto (em hexadecimal);
g) O valor da Tag (em hexadecimal).

2) Um computador possui uma memória principal com capacidade para 2 Gbits. Cada célula desta memória tem capacidade para 1 byte. Foi colocada neste computador uma memória cache associativa por conjunto com capacidade para 512 Kbytes. Cada linha desta cache tem capacidade para 16 células. Cada conjunto possui 4 linhas. Supondo que a CPU faça um acesso ao endereço $(02A7\ 4DB5)_{16}$, calcule: a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para o número do conjunto;
d) O total de bits para a Tag;
e) O número da coluna (em hexadecimal);
f) O número do conjunto (em hexadecimal);
g) O valor da Tag (em hexadecimal).

3) Um computador possui uma memória principal com capacidade para 2 Gbits. Cada célula desta memória tem capacidade para 2 bytes. Foi colocada neste computador uma memória cache associativa por conjunto com capacidade para 1 Mbyte. Cada linha desta cache tem capacidade para 512 bits. Cada conjunto possui 4 linhas. Supondo que a CPU faça um acesso ao endereço $(06ED\ C8AD)_{16}$, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para o número da conjunto;
d) O total de bits para a Tag;
e) O número da coluna (em hexadecimal);
f) O número da conjunto (em hexadecimal);
g) O valor da Tag (em hexadecimal).

4) Um computador possui uma memória principal com capacidade para 8 Gbits. Cada célula desta memória tem capacidade para 2 bytes. Foi colocada neste computador uma memória cache associativa por conjunto com capacidade para 1 Mbyte. Cada linha desta cache tem capacidade para 16 células. Cada conjunto possui 2 Kbits. Supondo que a CPU faça um acesso ao endereço $(1A5B\ CF7A)_{16}$, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para o número do conjunto;
d) O total de bits para a Tag;

e) O número da coluna (em hexadecimal);
f) O número do conjunto (em hexadecimal);
g) O valor da Tag (em hexadecimal).

5) Um computador possui uma memória principal com capacidade para 16 Gbits. O Barramento de Endereços deste computador possui 30 bits. Foi colocado nele uma memória cache associativa por conjunto com capacidade para 1 Mbytes. Cada linha desta cache tem capacidade para 512 bits. Cada conjunto tem capacidade para 128 células. Supondo que a CPU faça um acesso ao endereço $(0367\ 4AED)_{16}$, calcule:
a) O total de bits do endereço;
b) O total de bits para o número da coluna;
c) O total de bits para o número do conjunto;
d) O total de bits para a Tag;
e) O número da coluna (em hexadecimal);
f) O número do conjunto (em hexadecimal);
g) O valor da Tag (em hexadecimal).

3.3.2 Memória Somente de Leitura

A ROM (read-only memory), é um circuito de armazenamento apenas capaz de leitura. Seu conteúdo é permanentemente gravado pelo seu fabricante durante sua produção, devido a isso ele não pode ser alterado perante seu acesso, exceto por meios especiais. Diferente da memória RAM, seus dados não são perdidos quando a alimentação do sistema é interrompida, dado que é ele que guarda a informação da BIOS, responsável pela inicialização do sistema (boot).

A memória ROM é encontrada em celulares, cartuchos de videogames, impressoras e até em satélites, dado o funcionamento do firmware, um software imbutido diretamente no hardware, que guarda funções de instruções no hardware desses aparelhos, comparável com um sistema operacional.

BIOS, sigla de Basic Input/Output System (Sistema Básico de Entrada/Saída), é um firmware não volátil localizado físicamente na placa-mãe. Ele é responsável pelo processo de boot do sistema operacional, verificando e inicializando todos os componentes básicos do computador (processador, unidades de entrada e saída, drives, entre outros).

Outro firmware importante é o CMOS, abreviatura para Complementary Metal Oxide Semiconductor, que é uma pequena área de memória volátil encarregada por armazenar as configurações do setup (interface gráfica de acesso aos dados da BIOS) para que elas não sejam perdidas perante o desli-

gamento do sistema. No processo de boot a BIOS lê as informações do setup e opera segundo elas. Por ser volátil, ela é alimentada constantemente por uma bateria removível acoplada a placa-mãe que tem vida útil entre 2 a 3 anos. Essa técnologia também é usada na fabricação de circuitos integrados.

Existem três tipos principais de memória ROM, elas são:

- a PROM (Programmable Read-Only Memory);
- a EPROM (Erasable Programmable Read-Only Memory), e;
- a EEPROM (Electrically-Erasable Programmable Read-Only)

A PROM é uma memória produzida sem informações armazenadas em que a gravação de dados é efetuada com uso de correntes elétricas, onde os bits que antes estavam no estado 1, são alterados para 0. Neste tipo de ROM a gravação é somente possível uma única vez, não podendo alterar nem remover os dados posteriormente.

A EPROM, diferente da PROM, é uma memória apagável, nela existe a possíbilidade de remoção dos dados. A regravação é feita por exposição da memória á uma forte luz ultravioleta, onde o processo dura em média 20 minutos e apaga completamente todos os dados anteriores.

Por último, a EEPROM, ao contrário das anteriores, pode ser reprogramada diversas vezes. Nela a regravação pode ser parcial ou total, sendo realizada eletricamente dentro do próprio circuito. A memória Flash utilizada em cartões de memória e Pen Drives é um dispositivo moderno não volátil baseado na memória EEPROM, capaz de realizar as mesmas operações, mas possuindo algumas vantagens como maior durabilidade e densidade de armazenamento de dados.

3.3.3 Exercícios

1) (DOMCINTRA 2010 Câmara Municipal de Petrópolis) É a finalidade da memória cache:
A) permitir o boot pelo CMOS;
B) aumentar a área de backup da memória ROM;
C) acelerar o processamento do sistema;
D) permitir a utilização de resolução de vídeo 640 x 480;
E) garantir a utilização do “plug and play”.

2) Em qual tipo de memória ficam armazenadas as configurações do setup da placa-mãe?

A) CMOS B) RAM C) CACHE D) ROM E) DDR

3) (FGV 2007 SEFAZ MS) Na memória ROM dos microcomputadores, os fabricantes gravam um firmware que tem por objetivo realizar um autoteste na máquina quando esta é ligada, sendo executadas rotinas para identificação da configuração, inicialização dos circuitos da placa-mãe e do vídeo, testes de memória e teclado e carga do sistema operacional para a memória RAM. Esse programa é chamado de:
a) SETUP. b) BOOT. c) LOAD. d) BIOS. e) POST.

4) (FCC 2009 DPE/SP) Os cartões de memória, pendrives, memórias de câmeras e de smartphones, em geral, utilizam para armazenar dados uma memória do tipo:
A) FLASH. B) RAM. C) ROM. D) SRAM. E) STICK

3.4 Memória Externa

3.4.1 Memória Secundária

A memória secundária, ou memória de massa, é um tipo de memória com capacidade de armazenamento relativamente superior aos outros tipos e, diferentes da maioria, não é volátil, isto é, não perde seu conteúdo perante o desligamento do sistema. Como ilustrado na pirâmide de hierarquia de memória, a memória secundária ocupa a posição mais baixa em virtude de sua baixa velocidade de acesso que é consequencia de seu acesso indireto ao processador, isto é, para acessar o processador, a memória secundária primeiramente necessita passar pela memória principal, em contrapartida se comparada aos outros tipos de memória, ela possui um menor custo por byte armazenado.

4

Capítulo 4 - Entrada e Saída

Nesse cápitulo abordaremos os módulos de entrada e saída que são ligados a memória secundária, eles são os módulos responsáveis pela comunicação lógica com os dispositivos externos ao sistema computacional, eles tem acesso de ida e volta aos barramentos. Eles são usados por conta da variedade de periféricos, aparelhos que recebem ou enviam informações ao computador, como impressoras e leitores de CD e DVD, que possuem mecanismos de funcionamento distintos.

4.1 Módulos de Entrada e Saída

Os periféricos geralmente utilizam de formatos de dados e tamanhos de palavras diferentes dos dispositivos internos da máquina, cada um dispondo de um modelo que cumpra sua função, o que impossibilita o desenvolvimento de um processador capaz de controlar todos dispositivos externos. Esses aparelhos possuem uma taxa de transferência de dados relativamente baixa, o que faz desnecessária a utilização de barramentos (de dados, de endereços ou de contole) de alta velocidade ante a uma interação direta a esses dispositivos.

Os módulos de E/S possuem cinco funções, são elas o(a):

- Controle e Temporização;
- Comunicação com o microprocessador;
- Comunicação com o dispositivo;
- Área de armazenamento temporário de dados;
- Detecção de erros.

Controle e Temporização

A função é responsável pela verificação do estado do dispositivo por meio de uma investigação do módulo de E/S, que retorna o estado do dispositivo. Caso o retorno do estado indique que o aparelho esteja pronto para transmissão de dados, o processador solicitará a transferência para o módulo de E/S, que irá receber uma unidade de dados do dispositivo e entregar-lo ao processador.

A comunicação com o microprocessador

Ele tem papel de decodificador de instruções, realizando a tradução dos dados transferidos pelo barramento de controle, o barramento que controla tanto o barramento de dados (o transporte de dados, ele também realiza o envio de dados entre o processador e o módulo) quanto o de endereços (designador de endereços de memória). O módulo também reconhece um endereço diferente para cada dispositivo conectado.

Em consequência da baixa taxa de transferência de dados do periférico, é relevante saber o estado do módulo de E/S para realizar as operações.

Comunicação com o dispositivo

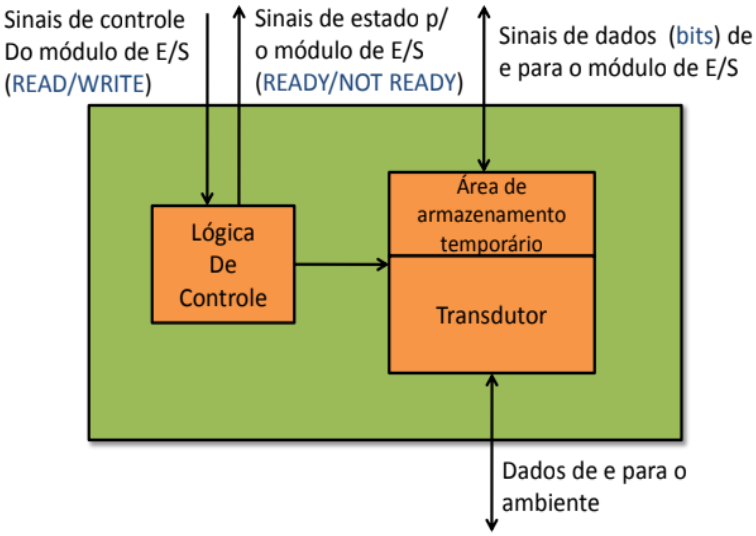


Figura 4.1: Ilustração da comunicação com o dispositivo retirada do módulo de Arquitetura de Computadores de Helcio Wagner da Silva.

Na imagem acima é exibido o funcionamento da comunicação do dispositivo com o computador. Nela o módulo de E/S envia ou recebe sinais de controle, sinais que determinam como o disposivo deve agir (desempenhar uma função, informar estado, entre outros). A lógica de controle do dispositivo tem como função realizar uma operação baseada no comando recebido do módulo de E/S.

O transdutor faz a conexão entre o dispositvo e o ambiente externo, convertendo dados decodificados em sinais elétricos para outras outras formas de energia e vice-versa. A ele é integrado uma pequena área de armazenamento volátil para auxílio na transferência de dados.

Área de armazenamento temporário de dados

Funciona exatamente como seu nome indica, ela guarda os dados transferidos da memória principal para o módulo de E/S, um processo que é realizado bastante rápido, para em seguida transferir-los para o periférico numa velocidade correspondente a seu padrão. No trajeto de volta, os dados são armazenados temporariamente no módulo de E/S para que não retenham a memória em uma transferência de dados a baixa velocidade. O módulo é capaz de realizar as operações á duas velocidades diferentes, a da memória e a do periférico.

Detecção de erros

Ela detecta os erros que podem ser divididos em dois tipos: físicos ou lógicos. Eles são, respectivamente, os erros gerados por mau funcionamento elétrico ou mecânico (falha de alimentação, trilha de disco defeituosa, entre outros) e alterações no padrão de bits enviados pelo dispositivo externo.

Técnicas de Transferência

Existem três técnicas usadas nos módulos de E/S para transferência de dados: a programada, a dirigida por interrupção e a de acesso direto a memória (DMA), com as duas ultimas sofrendo de interrupções.

	Sem Interrupções	Com Interrupções
Transferência entre memória e E/S por meio do µP	E/S programada	E/S dirigida por interrupção
Transferência direta entre memória e E/S	-	Acesso Direto à Memória (DMA)

Figura 4.2: Imagem ilustrando as técnicas de transferência oriunda do módulo de Helcio Wagner da Silva.

E/S programada

Na E/S programada uma execução de uma instrução relacionada a E/S retorna um comando para o módulo dela, esse comando é executado pelo módulo que indica que seu processo é concluído quando carrega um valor em seu registrador de estado. Esse módulo não irá alertar o microprocessador que a operação terminou, é responsabilidade do processador investigar o estado dos módulos frequentemente a fim de conferir o término de operações.

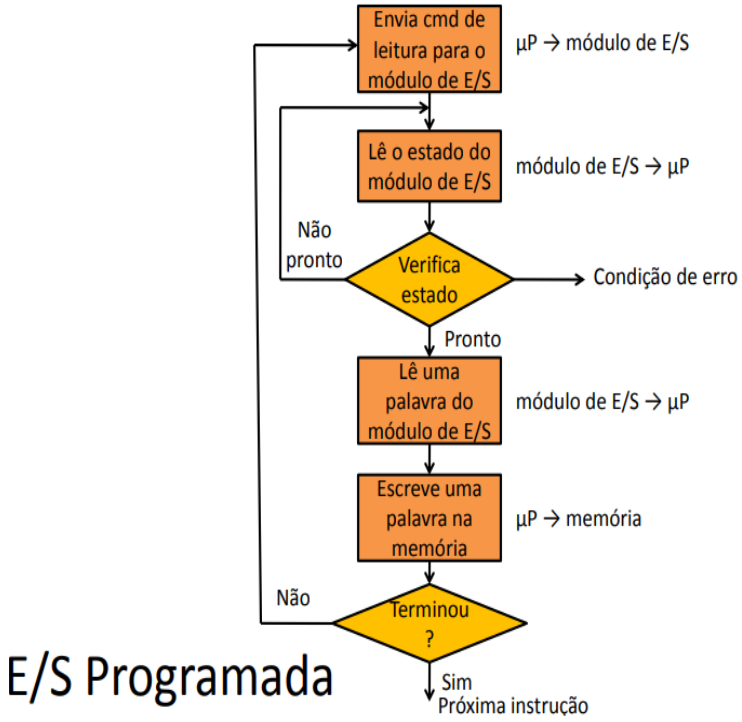


Figura 4.3: E/S programada. Fonte: módulo de Helcio Wagner da Silva.

Na transferência programada existem duas formas de endereçamento de dispositivos: a E/S mapeada em memória e a E/S independente. Na mapeada em memória, existe apenas um espaço de endereçamento usado tanto

para as posições de memória quanto para os dispositivos de E/S, o que tem vantagens e desvantagens dependentes do espaço de memória. Ela também compartilha as instruções de acesso á memória e ao dispositivo. Na E/S independente, os espaços de endereçamento são divididos, o E/S possui um e a memória, outro.

E/S dirigida por interrupção

Nesse tipo de transferência, diferente da programada, onde o processador tem papel de verificador, o processador envia um comando para o módulo de E/S e continua a realizar outras operações enquanto espera pela resposta do módulo. Quando estiver pronto para a troca de dados, o módulo interrompe os procedimentos do processador, dando pausa as operações não inicializadas, para realizar a transferência de dados que, quando acabada, permite que o processador retorne a suas atividades. Em razão da eliminação de tempo de espera desnecessário, esse método se mostra mais eficaz que o programado.

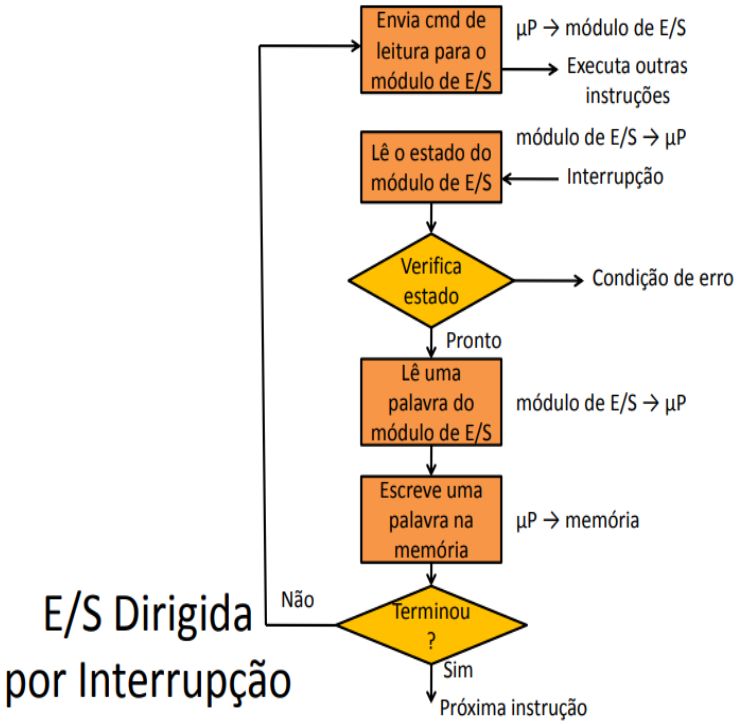


Figura 4.4: E/S dirigida. Fonte: módulo de Helcio Wagner da Silva.

E/S de acesso direto a memória

No Acesso Direto à Memória é utilizado de um módulo extra, chamado de Controlador de DMA, no barramento do sistema que permite a transferência direta de dados entre o periférico e a memória. Esse módulo funciona como uma imitação do processador, executando todas suas funções. Ele é útil em casos onde a transferência de dados entre memória e dispositivo é grande, que é uma situação onde o uso do método de interrupção é ineficaz.

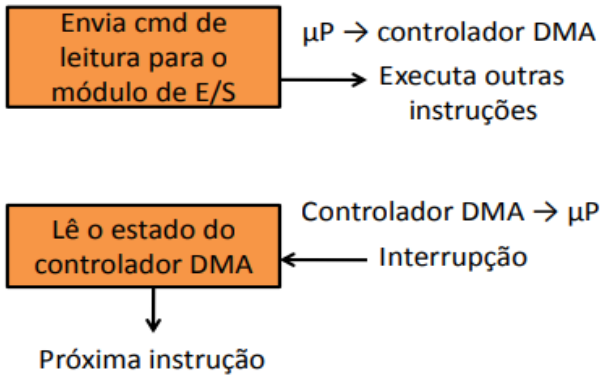


Figura 4.5: Processo de E/S DMA. Fonte: módulo de Helcio Wagner da Silva.

O controlador de DMA opera de dois modos: ele utiliza do barramento quando o processador não o esta usando ou força o processador a suspender temporariamente sua operação (roubo de ciclo).

Para DMA temos diversos tipos de Barramento:

- Barramento único, DMA separado: Custo baixo, mas ineficiente em consequencia dado que a transferência de cada palavra consome vários ciclos do barramento;
- Barramento único, DMA-E/S integrados: Custo alto, mas eficiente;
- Barramento específicode E/S: Custo alto, mas eficiente.

4.1.1 Exercícios

1) Qual função dos módulos de E/S está relacionada ao compartilhamento de recursos, tais como o barramento e a memória principal, pelas várias atividades que são realizadas por um sistema?
a) Armazenamento temporário dos dados.
b) Comunicação com dispositivos.
c) Comunicação com o processador.
d) Detecção de erros.

e) Temporização.

2) A principal função do transdutor em um Módulo de E/S do computador é:
a) determinar a função a ser executada pelo dispositivo.
b) indicar o estado do dispositivo.
c) armazenar os dados em uma área temporária para serem transferidos.
d) controlar a operação de um dispositivo.
e) converter os dados codificados como sinais elétricos para alguma outra forma de energia ou vice-versa.

3) Diferencie as técnicas utilizadas para E/S (programada, interrupção e DMA), considerando o envolvimento e ações do processador x impedimento do processador e como ficaria a multiprogramação em cada um desses. Qual deles tem um comportamento chamado de Polling? (Roubo de ciclo) Em quais momentos do ciclo de instrução da CPU é possível utilizar a DMA?

4) Parte da definição da arquitetura de um computador é a especificação do seu sistema de entrada/saída. O esquema de E/S no qual a CPU gasta maior parte do seu tempo em loop, esperando o dispositivo ficar pronto, é a E/S:
a) Por interrupção.
b) DMA.
c) Programada.
d) Através de canais.
e) Por barramento.

5) Qual o processo que verifica periodicamente o status de um dispositivo de E/S para determinar a necessidade de atender ao dispositivo:
a) Polling.
b) Instruções de Entrada/Saída.
c) Handshaking.
d) Entrada/Saída controlada por interrupção.

6) Qual das alternativas abaixo é função dos módulos de entrada e saída:
a) Controle e temporização.
b) Armazenamento contínuo de dados.
c) Comunicação direta com a Unidade Lógica Aritmética.
d) Sincronização das trilhas de setores dos discos ópticos.

7) No contexto da organização de computadores, existem diversas formas de se realizar operações de Entrada e Saída (E/S) entre a Unidade Central de Processamento (UCP) e os dispositivos de E/S. Em uma dessas formas de E/S, a transferência de dados ocorre diretamente entre o dispositivo de E/S

e a memória principal do computador, sem a interferência da UCP. Trata-se da transferência denominada:
a) Multithreading.
b) E/S Programada.
c) E/S por Amostragem.
d) E/S por Interrupção.
e) Acesso Direto à Memória (DMA).

4.2 Memória Secundária e Calculos de Acesso ao Disco com Alocação Sequencial e Aleatória

O disco possui um tempo de acesso que é o somatório de três medidas que levam para realizar o acesso a um setor do disco, essas medidas sáo o tempo de busca (seek time), o atraso rotacional e a taxa de transferência.

Seek Time

É o tempo de atraso gasto para a cabeça de leitura e gravação se mover físicamente até o local apropriado. O processo de busca é chamado de *seeking* e o tempo gasto pelo cabeçote é o *seek time* (tempo de busca). Dependendo de onde se encontra físicamente o cabeçote o seek time pode variar, em função disso, é usado o tempo de busca médio que é o tempo de busca dividido pela metade.

Atraso rotacional

Atraso rotacional ou latência de rotação é o tempo médio levado para realizar uma rotação completa pelo disco. Ele é calculado assim dado que o setor específico do disco pode ter acabado de passar sob a cabeça quando foi requisitado.

Taxa de transferência

É o tamanho médio de dados por unidade de tempo que passam entre dispositivos num computador.

Abaixo se encontra a fórmula de tempo de acesso.

$$T_A = T_S + 1/2 * T_R + T_T$$

Em que T_A equivale a tempo de acesso, T_S ao tempo de busca, T_R ao atraso rotacional e T_T ao tempo de transferência.

Exemplo:

Um arquivo possui 4000 blocos. Deseja-se acessar os dados armazenados nos 1500 primeiros blocos. Calcule o tempo de acesso, considerando seguir a alocação:
a) sequencial
b) aleatória
Os setores são de 512 bytes, uma trilha possui 750 setores. O disco possui uma velocidade de rotação de 6000 rpm e tempo médio de busca é igual a 10 ms.

Primeiramente íremos calcular pelo método sequencial, onde o tempo de busca é de 10ms para a primeira trilha e 0 para as ademais. Considerando que o atraso rotacional possui uma velocidade de 6000 rpm, podemos transforma-lo para segundos simplesmente dividindo o por 60:

$$1/RPM/60 = 1/6000/60 = 1/100 = 10ms$$

Disso também podemos inferir que o atraso rotacional médio é de 5 ms.
Por último, o tempo de transferência é calculado por: $b*r/N$, onde b equivale ao número de bytes transferidos, r a quantidade de tempo, em segundos, para uma rotação e N ao número de bytes por trilha. Como cada trilha possui 750 setores de 512 bytes cada e a quantidade de blocos buscada é de 1500, teremos que calcular o tempo de acesso da primeira e segunda trilhas. O calculo será o seguinte:

$$b * r/N = 750 * 512/750 * 512 = 10ms$$

Com o valor dos três tempos de atrasos podemos descobrir os tempos de acesso, que serão:
Primeira trilha:$T_A = T_S + 1/2 * T_R + T_T = 10 + 5 + 10 = 25$ms
Segunda trilha: $T_A = T_S + 1/2 * T_R + T_T = 0 + 5 + 10 = 15$ms

No método aleatório, por outro lado, o tempo de acesso sempre será calculado como de 10ms devido ao espalhamento aleatório dos setores no disco enquanto o atraso rotacional é o mesmo do método sequencial. O calculo de tempo de transferência também varia, em vez de calcularmos por trilha, a operação será feita por setores como a expressão a seguir mostra:

$$T_T = b * r/N = 10 * 1 * 512/750 * 512 = 1/75 = 0,0133...ms$$

Onde 10 é o número de rotações por segundo, 1 é o número de setores e 512 o número de bytes transferidos. Descoberto o tempo de transferência, o tempo de acesso por setor será igual á:

$$T_A = 10 + 5 + 0,0133... = 15,0133...ms$$

Para achar o tempo de acesso total, basta multiplicar o tempo de acesso por setor pela quantidade de blocos procurados que, no caso, são 1500.

$$T_A = 15,0133... * 1500 = 22520ms$$

Exercícios

1) Um arquivo possui 4800 blocos. Deseja-se acessar os dados armazenados nos 1500 primeiros blocos. Calcule o tempo de acesso, considerando seguir uma alocação:
a) sequencial
b) aleatória
Os setores são de 512 bytes, uma trilha possui 600 setores. O disco possui uma velocidade de rotação de 12000 rpm e tempo médio de busca igual a 8 ms.

4.2.1 Discos Rígidos

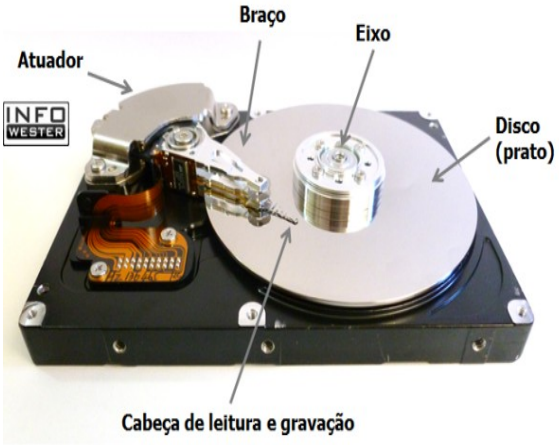


Figura 4.6: Interior do HD, fonte: https://www.infowester.com/hd.php

No disco rígido, ou HDD (Hard Disk Drive), os discos magnéticos são pratos circulares, criados a partir de materiais não magnéticos (essa camada é denominada de substrato), compostos de uma cobertura fina de material magnetizável da qual sua densidade de gravação é diretamente proporcional a espessura dessa camada magnética, quanto mais fina a camada maior será sua sensibilidade e maior será a densidade de gravação.

O material do substrato geralmente é o alumínio, a liga de alumínio ou, mais recentemente, o vidro. O vidro, em particular, possui algumas

vantagens: melhor rigidez, maior resistência a choques e danos, redução significativa nos defeitos da superfície e melhoria na uniformidade da superfície, aumentando a confiabilidade do disco.

O disco possui um cabeçote de leitura e escrita que, na verdade, é controlado por uma bobina com um eletroimã minúsculo dentro do atuador capaz de criar campos mágneticos no disco. Na escrita, a bobina realiza movimentos de correntes em sentidos opostos que criam padrões mágneticos distintos que representam sequências de 1 e 0 no disco. Na leitura, esse campo mágnetico induz uma correnta na bobina que determina o valor do bit lido.

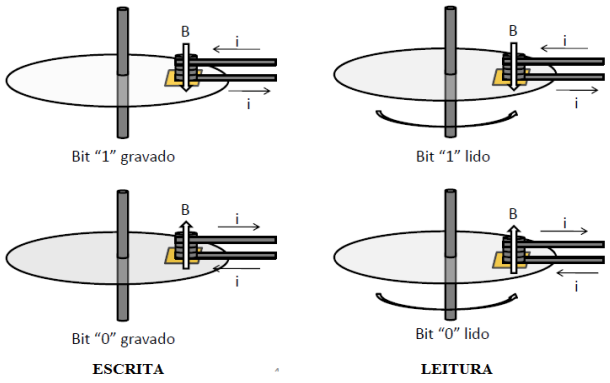


Figura 4.7: Leitura e escrita do disco.

A organização dos dados no prato é divido em partes: trilhas, lacunas e setores. Trilha é um conjunto concêntrico de anéis que quando adjacentes são separadas por lacunas para evitar requisitos de precisão excessivos no sistema e o setor é a subdivisão de uma trilha, no qual existem centenas de setores dentro dela. A transferência dos dados de e para o disco é feita em setores, que podem ter tamanho variável, mas que geralmente é fixa em 512 bytes para discos magnéticos.

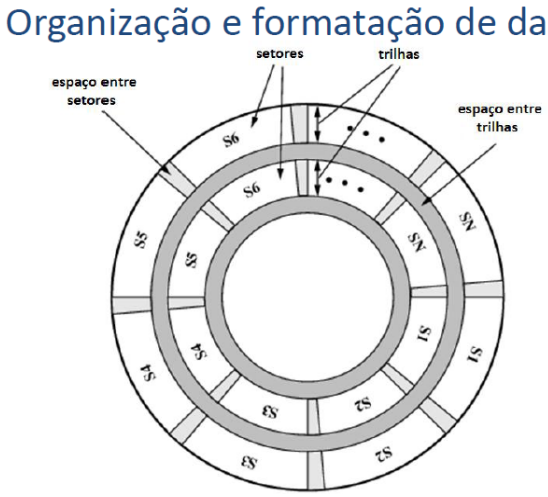


Figura 4.8: Arquitetura do disco, imagem do livro de William Stallings.

O disco agrupa os setores numa entidade chamada cluster que é a menor unidade de espaço alocável do disco, o cluster é endereçado pelo sistema que o reconhece como uma unidade lógica. O tamanho do cluster é diretamente proporcional a capacidade do disco, em razão de mais setores formarem ele. O cluster é indiferente ao tamanho do arquivo e sempre será usado pelo menos um para armazena-lo.

Existem dois métodos de leitura, cada um com suas vantagens e desvantagens:

O modo de velocidade angular constante (CAV) gira o disco numa velocidade fixa, lendo as informações em uma mesma taxa. Nela os setores tem formato de fatia de torta e as trilhas são concêntricas, onde cada trilha e cada setor tem unidades individuais endereçáveis. Uma desvantagem é que ela tem uma menor densidade de dados devido a perda de espaço nas trilhas externas que são acessadas a uma velocidade maior do que as internas.

O outro modo é o de velocidade linear constante, ou gravação em múltiplas zonas, usado nos primeiros drives de CD-ROM. Nesse modo, a velocidade de rotação fazia o oposto do que acontece na CAV, a velocidade era maior nas trilhas externas e menor nas internas.

O cabeçote de leitura é outro componente importante do disco, do qual os movimentos variam em fixos e móveis. No fixo há uma cabeça por trilha, em que cada uma é montada sobre um braço rígido fixo, e no móvel, uma por superfície, montada sobre um braço móvel.

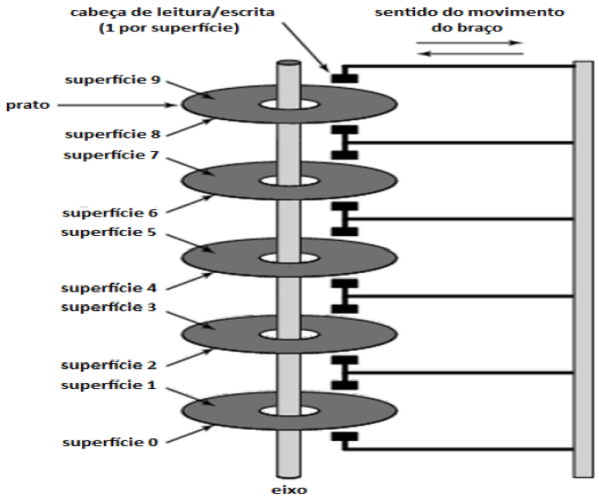


Figura 4.9: Movimento do cabeçote, imagem do livro de William Stallings.

Há a transportabilidade do disco, que pode ser removível ou não-removível. O removível pode ser substituído por outros, possui facilidade de transferência de dados entre sistemas diferentes e dispõe capacidade de armazenamento ilimitada, enquanto o não-removível não detém destas características por ser montado permanentemente a unidade.

Num disco podem ser usados múltiplos lados, cada qual com uma cabeça que, por todas estarem vinculadas a um mesmo braço, sempre movem juntas, o que torna impossível existirem cabeças em trilhas diferentes. O alinhamento vértical dessas trilhas recebe o nome de cilindro onde dados são espalhados, reduzindo o movimento da cabeça e aumentando a taxa de transferência de dados.

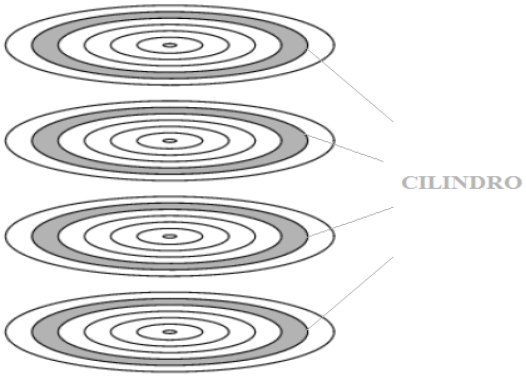


Figura 4.10: Cilindro.

Tecnólogias de Discos Rígidos

A busca por maior velocidade e desempenho nas máquinas gerou diferentes tipos de tecnólogias voltadas para o HD, cada qual com suas vantagens e desvantagens. Nessa subseção iremos expor alguns padrões para interfaces de controladores do disco rígido, que são as unidades responsáveis pelos dispositivos de armazenamento de dados.

IDE

Integrated Drive Electronics foi uma interface que surgiu durante a década de 80, ela foi a primeira tecnologia com controlador integrado ao disco rígido. Seu endereçamento de setores era dado pela quantidade de cabeçotes, cilindros e setores, convenção denominada de CHS (Cylinder, Head, Sector). A necessidade de mais cilindros para um endereçamento correto foi um dos problemas dessa tecnologia. Seus padrões foram IDE (ou ATA 1), EIDE (ou ATA 2), ATA 3, ATA/ATAPI 4, ATA/ATAPI 5, ATA/ATAPI 6, ATA/ATAPI 7 e ATA/ATAPI 8.

SATA

Serial Advanced Technology Attachment ou Serial ATA foi um modelo de disco que substituiu o IDE. Concebido em 2003, ele utilizou do conceito de sequências, como seu nome indica, para atingir maior velocidade de transmissão a partir de frequências maiores. Ele engloba os padrões SATA 150 (ou I), SATA II, SATA 300 (ou SATA/300) e SATA 600.

SCSI

Small Computer System Inferface foi inventanda em 1979 por Howard Shugart e padronizada em 1986 pela ANSI. SCSI foi, além de uma interface, um barramento que suportava a ligação de vários periféricos, conectando até 16 dispositivos (cada um recebendo um ID de 0 a 15). As altas taxas de transferências fizeram dela o padrão da época. Seus padrões foram SCSI 1, SCSI 2 e SCSI 3.

SAS

A tecnologia Serial Attached SCSI utiliza dos comandos do padrão SCSI em sequências, obtendo maior rapidez, desempenho, escabilidade (manipulação uniforme em escala crescente) e gerenciamento na transmissão de dados. Ela possui compatibilidade com modelos SATA e suporta 4 dispositivos por cabo.

4.2.2 Solid State Drive

Devido as limitações tecnológicas do HDD (Hard Disk Drive), que usa de cabeçotes para leitura e gravação, partes que restringem a velocidade que o dispositivo opera, muitas vezes acarretando que o processador e a memória aguardem pela chegada de dados do HD, surgiu a necessidade de uma tecnólogia que removesse essa limitação. A solução para esse problema manifestou-se na forma do SSD, ou unidade de estado sólido, uma tecnólogia de armazenamento não volátil.

Diferente do HD, o SSD não tem partes móveis como o cabeçote de leitura e gravação, em vez disso, o SSD é composto por uma memória Flash e um controlador. A memória Flash, além de armazenar arquivos, realiza as operações de leitura e gravação eletricamente, ocasionando que elas sejam mais rápidas que a do HD e que o dispositivo seja mais silêncioso e resistente á choques físicos. O controlador, que é uma espécie de processador, tem papel de gerenciar a transfêrencia de dados entre o HD e o computador, criptografando informações, administrando as operações de leitura e escrita, detectando e corrigindo erros, entre outras tarefas. Ele também garante maior vida útil a memória Flash.

Algumas vantagens dos SDD são:

- Menor tempo de acesso, em razão das operações de leitura e gravação serem por meio elétrico;
- A remoção das partes móveis e magneticas faz com que o dispositivo seja silencioso e mais resistente;
- Mais leve e menor por possuir menos componentes físicos que o HD;
- Menor temperatura e consumo de energia;
- Maiores taxas de transferência.

As desvantagens são:

- Maior custo se comparado com os HDs;
- Capacidade de armazenamento inferior á alguns tipos de dísco rigido (IDE, SATA).

Um ponto que pode ser considerado como uma desvantagem por alguns é quanto ao tempo útil da memória Flash, que depende da quantidade de ciclos de escrita, diferente do HD que, enquanto tende a sofrer defeitos mecânicos, não possui essa limitação. Esse problema é desconsiderado atualmente graças a vida útil longa dos SSD modernos e das técnicas de distribuição de dados, como a *wear leveling* (distribuição de uso), que permite o controlador

gravar dados em blocos (agrupamento de páginas, a menor parte física do SSD) pouco utilizados.

Outra característica que os SSD produzidos atualmente possuem é a possibilidade de evitar deixar o drive lento. O sistema operacional usa do comando TRIM para limpar blocos de arquivos removidos, retornando o SSD a um estado praticamente novo. Uma desvantagem consequente desse comando é a inviabilidade da recuperação de dados deletados, o que afeta principalmente o campo da Ciência Forense.

O SSD também pode ser desfragmentado á risco de diminuição da vida útil do dispositivo, o que torna essa ação não recomendável dado que não há necessidade alguma de desmonta-lo.

Exercícios

1) Com um disco organizado com setores de 512 bytes, uma trilha com 1024 setores e velocidade de rotação de 6000 RPM, para acessar um arquivo com 1280 blocos.
a) Calcule o tempo de acesso total considerando alocação sequencial.
b) Calcule considerando alocação aleatória.

2) Um arquivo possui 4000 blocos. Deseja-se acessar os dados armazenados nos 1500 primeiros blocos. Calcule o tempo de acesso, considerando seguir uma alocação:
a) Sequencial
b) Aleatória

Os setores são de 512 bytes, uma trilha possui 750 setores. O disco possui uma velocidade de rotação de 6000 rpm e tempo médio de seek igual a 10 ms.

4.3 RAID

RAID é a abreviação de Redundant Array of Independent Disks, ou conjunto redundante de discos independentes em português, uma tecnologia desenvolvida com intuito de melhorar tanto o desempenho quanto a segurança dos discos rígidos, o Hard Disk (HD) ou, como foi denominado anteriormente, memória secundária, através da redundância de dados.

Esse mecanismo atua por meio da utilização de discos extras (dois ou mais discos) que trabalham em paralelo, dividindo suas informações para alcançar o mesmo objetivo. Caso ocorra erro em algum disco, os outros

continuaram a funcionar corretamente, o que não resultará em prejuízos excetuando situações de falta de alimentação de energia, erros operacionais ou códigos errôneos, que são prevenidos com o uso de backup.

A RAID originalmente detinha de 5 níveis que, depois de várias modificações, passaram a ser diversos, mas com apenas seis de uso comum. Neste livro iremos focar na RAID 0, 1, 4, 5, 6 e 10 (0+1 e 1+0) os níveis de RAID mais usados atualmente, citando brevemente outros níveis.

A RAID 0 (Espalhamento de Dados) utiliza do método de striping, distribuição de dados, em que o dísco lógico é dividido em tiras (strips) que são escritas sequencialmente nos discos disponíveis. Um conjunto de strips é conhecido como um stripe e seu tamanho é relacionado com seu desempenho. Esse método possui uma taxa de transferência e uma taxa de processamento de solicitações altas, mas nenhuma redundância (paridade).

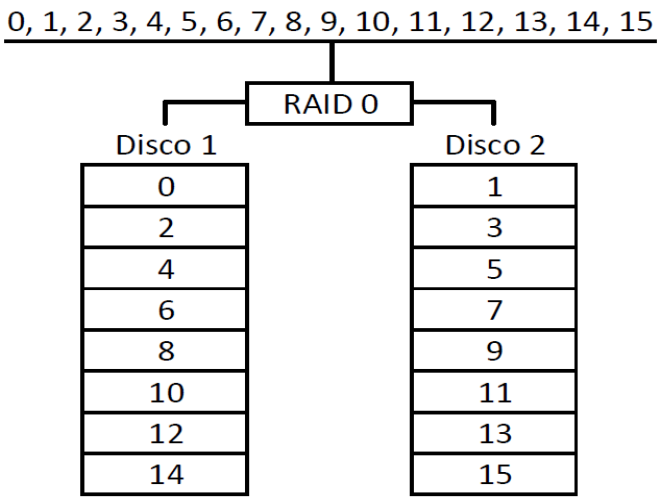


Figura 4.11: Raid 0.

A RAID 1 (Replicação de Dados) usa da técnica mirroring ou espelhamento de dados, nela todos os dados são escritos em todos os discos, caso haja perda ou remoção de um disco, o processo não é interrompido uma vez que outros discos possuem a mesma informação. Ele possui uma taxa de processamento de solicitações razoável, uma taxa de transferência alta e um custo alto se comparado a RAID 0. Essa RAID, como a RAID 1, não usa de paridade.

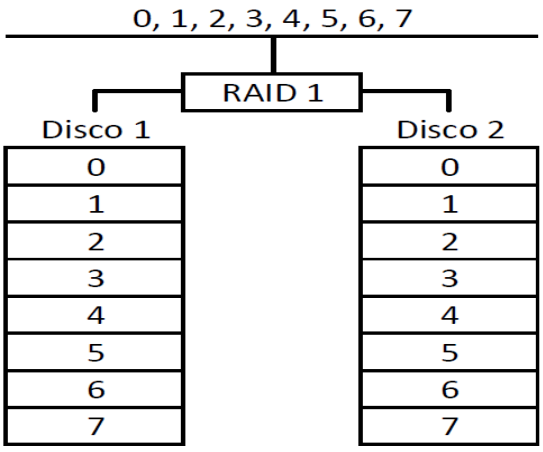


Figura 4.12: Raid 1.

Na RAID 4 (Paridade de Bloco) um dos discos guarda a paridade do bloco enquanto os outros são acessados independentemente e as operações ocorrem em paralelo. Caso haja falha em um disco, o funcionamento ficará lento até a troca do disco defectivo. Sua taxa de processamento de solicitações é alta e a de transferência é aceitável.

A RAID 5 (Paridade Distribuída) funciona como a 4, a diferença é que a paridade é distribuída em todos os discos em vez de apenas um, o que resulta em maior desempenho. A RAID 5 é composto por três ou mais discos e possui paridade distribuída, isto quer dizer que o RAID 5 não possui um disco dedicado para paridade, ela é gravada a cada momento em um diferente disco do conjunto, de forma a distribuir uniformemente o Input/Output entre os discos. O RAID 5 suporta a perda de um disco de cada lado do RAID 0 sem que ocorra perda de dados ou indisponibilidade do ambiente Ela tem taxa de processamento de solicitações alta e taxa de transferência aceitável.

Por ultimo, a RAID 6 utiliza de paridade dupla, calculando duas paridades e distribuindo elas em todos os blocos, igual a RAID 5, o que faz necessário o uso de pelo menos dois discos. Graças a essa distribuição, ela só sofre perda de dados quando três ou mais discos falham. Ela tem redundância alta, um custo elevado e escrita lenta.

Entre os outros níveis, a RAID 2 utiliza de redundância via Código de Hamming em que os discos rígidos guardam correções de erros em posições correspondentes, um método bastante útil em virtude de uma alta minimização da taxa de erros pelo uso do ECC, mas antiquado no mercado atual onde os discos já possuem essa técnologia internamente. A RAID 3 é bastante parecida com a anterior, apenas simplificada. Ela usa de redundância

via paridade, com um disco armazenando os bits de correção dos outros.

RAID com números em duas unidades são junções de RAID com unidades singulares. A RAID 10 (1 + 0) é a união dos níveis 1 (Mirroring) e 0 (Stripping), contendo 4 ou mais discos (sempre aos pares), onde encontram-se diversos RAID 1 sob um RAID 0, assim gravando os dados em cada diferente par de discos, uma vez em cada disco do par. Essa formatação permite o RAID 10 surportar a perda de metade do número total de discos, contanto que ela não ocorra nos dois discos do mesmo par. Possue maior desempenho que o RAID 1, por usar múltiplos discos, mas é o RAID de maior custo uma vez que sua área útil é igual a metade do somatório da área de todos os discos.

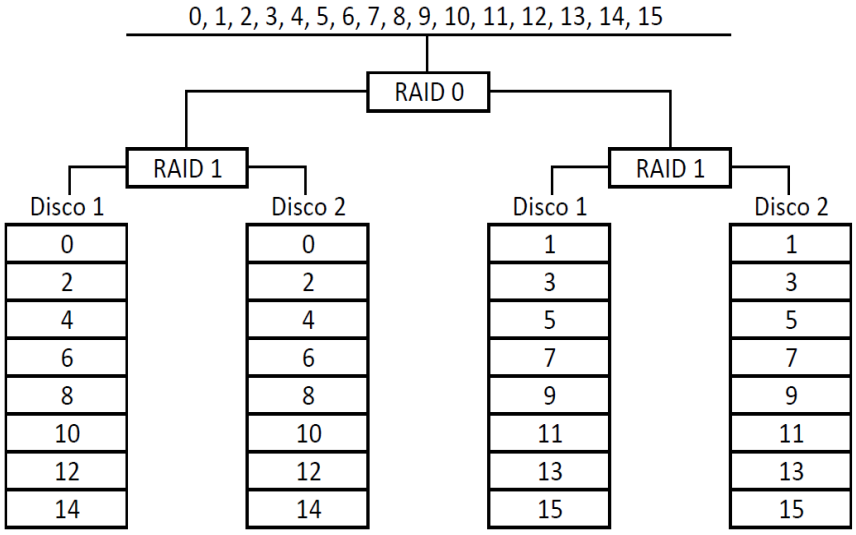


Figura 4.13: Raid 10.

Por outro lado, temos a RAID 01 que é outra combinação dos níveis 0 e 1. Nela também são utilizados no mínimo 4 discos em quantidade par, sendo que ela opera o oposto da RAID 10, gravando o mesmo dado em um par, cada qual tendo uma informação diferente. Caso haja falha em um dos discos, ela deixará de ser RAID 01 se tornará uma RAID 0

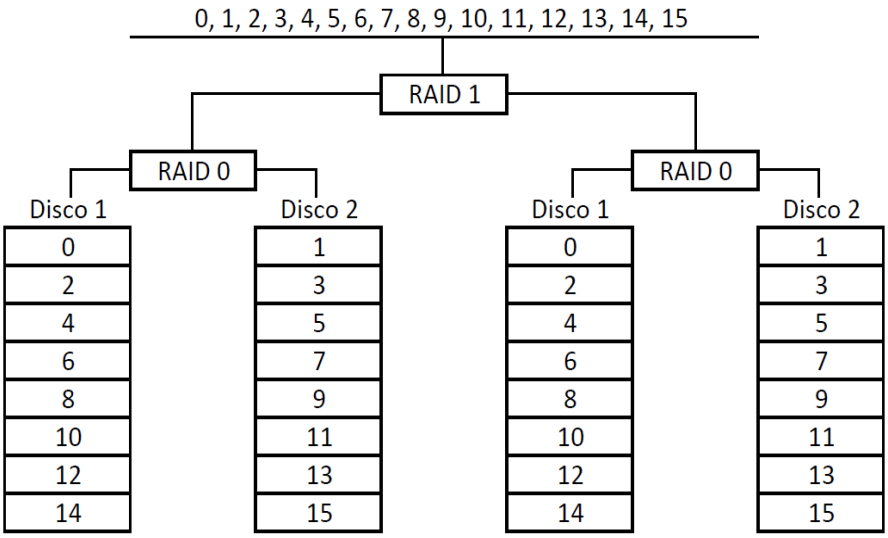


Figura 4.14: Raid 01.

Exercícios:

1) A tecnologia RAID (Redundant Arrays of Independent Disks) permite a um computador usar duas ou mais unidades de disco rígido ao mesmo tempo. A tecnologia RAID trata várias unidades como uma unidade contígua. Com relação a esta tecnologia, é correto afirmar:
a) É uma tecnologia que é implementada através de configurações de hardware ou software.
b) Soluções que implementam RAID 0, RAID 2 e RAID 5 oferecem tolerância a falha, suportando possíveis falhas nos discos rígidos.
c) O RAID 5 somente pode ser implementado tendo-se à disposição um mínimo de 5 dispositivos de armazenamento (discos rígidos).
d) Soluções de armazenamento que utilizam RAID 1 são baseadas em espelhamento dos dados, enquanto RAID 10 opera com a tecnologia e emprego de paridade aos dados armazenados.
e) No RAID 0 (Striping), os dados são divididos e armazenados em mais de uma unidade de disco física. Desta forma o desempenho de leitura/gravação é aprimorado, oferecendo também redundância para garantir recuperação na possível falha de dispositivo de armazenamento.

2) Usando a combinação de disco em RAID 50 (0 +5 - RAID 0 no topo e 5 em baixo) organize os blocos A, B, C, D, E, F, G, H de um arquivo nos discos necessários , mostrando os blocos que ficam repetidos nos discos.

3) Usando a combinação de disco em RAID 10 (0 + 1 - RAID 0 no topo e 1 embaixo) organize os blocos A, B, C, D, E, F, G, H de um arquivo nos

discos necessários, mostrando os blocos que ficam em cada um dos discos do RAID 1. No total devem ser usados 4 (quatro) discos.

4) O nível de RAID que implementa com, no mínimo, dois discos o espelhamento de disco para, em caso de problema com um deles, o outro possa manter a continuidade de operação do sistema, é o:
a) RAID 5 b) RAID 0 c) RAID 1 d) RAID 10 e) RAID 6

4.4 Barramento e interfaces

Os barramentos são conjuntos de linhas de comunicação que interligam dispositivos no computador, eles possuem três tipos de funções: de endereços, de dados e de controle.

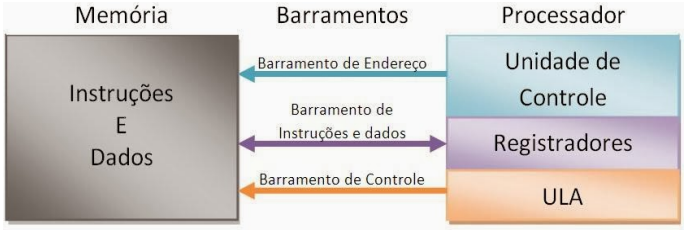


Figura 4.15: Ilustração dos barramentos no sistema, imagem do livro Elementos da Computação da Escola Técnica Municipal “Os Padres do Trabalho".

O barramento de dados tem como função transportar informações (dados e instruções) e é bidirecional. O barramento de endereços é utilizado pelo processsador para indicar endereços de memória ou dos dispositivos de E/S a serem retirados ou enviados, tem tamanho equivalente aos espaços de memória acessados e é unidirecional. Por ultimo, o barramento de controle controla os outros dois, confirmando e solicitando as ações a serem realizadas, ele é bidirecional.

O uso de barramentos possue suas vantagens, como a versatilidade ocasionada pelo uso de um único esquema de ligação que permite incorporar mais periféricos e movimenta-los entre diferentes máquinas. Outro benefício é o baixo custo que esse esquema possui quando distribuido entre vários aparelhos. Algumas desvantagens são a criação de um gargalo na velocidade de comunicação mediante o uso de vários dispositivos e o limite de velocidade de transmissão que é delimitado também pelos aparelhos ligados.

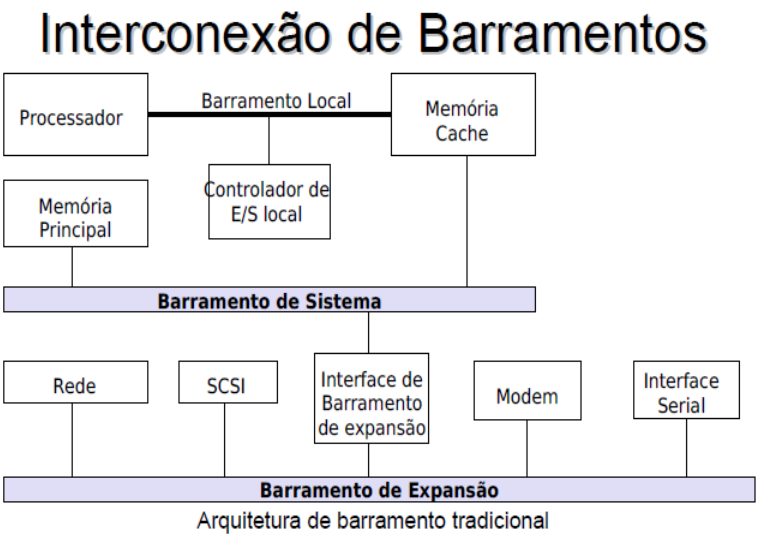


Figura 4.16: Interconexão de barramentos, imagem retirada do livro de William Stallings

Na imagem acima, nota-se a existência do barramento do sistema e do barramento de expansão, que possibilita a conexão á vários dispositivos de E/S, ambos presentes na arquitetura de barramento tradicional. Esse tipo de arquitetura é eficiente, mas não adequada para a interligação á dispositivos de E/S mais modernos em razão de maior demanda de desempenho para funcionamento regular. A solução para esse transtorno é utilizar barramentos de alta velocidade, como mostra a figura abaixo:

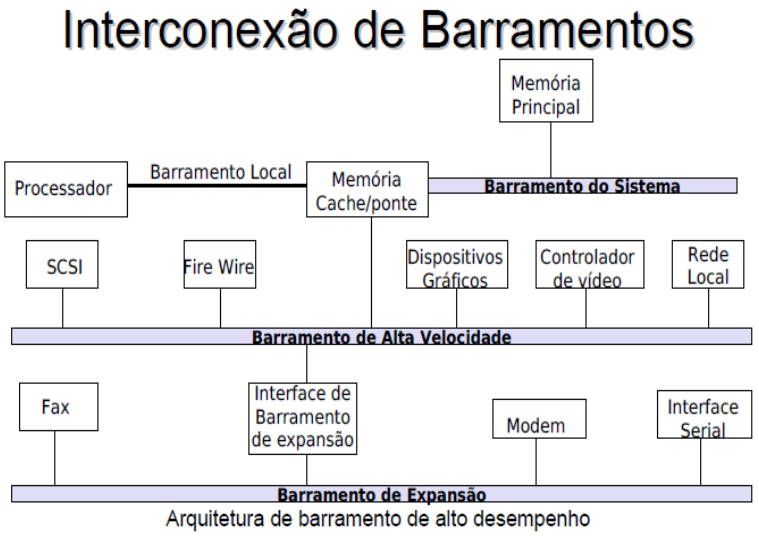


Figura 4.17: Interconexão de barramentos de alta velocidade, imagem retirada do livro de William Stallings

Nela, o barramento de alta velocidade é independente do processador e permite maior número de ligações a periféricos com maior demanda de desempenho.

5

Capítulo 5 - Tópicos avançados

5.1 Pipeline

Pipeline se refere a técnica de segmentação de instruções que é a divisão das etapas do processamento de instruções. Num ciclo de instrução com uso de pipeline, quando o CPU está executando uma instrução na ULA, ele já esta decodificando a seguinte e buscando mais outra, ganhando velocidade de processamento.

Essa divisão de etapas se torna complexa por consequência de desvios de instrução. Por exemplo, na máquina de Glenn existe o código de operação B que, caso a condição seja verdadeira, o PC irá saltar para a posição de memória esperada, fazendo com que o CPU perca as informações de decodificação e execução das instruções já guardadas, desaproveitando o progresso já realizado. O mesmo acontece na operação D, em que o CPU terá que parar seus processos para acessar a posição do pixel desejado.

Outros problemas provêm do diferente grau de complexidade de instruções, da qual algumas levam mais tempo que outras para serem executadas, e da dependência de instruções, em que a próxima depende da anterior.

A imagem abaixo simula o uso de pipelines na máquina de Glenn.

Posição de Memória	Busca	Decodifi-cação	Execução
C0	Instrução 1		
C1			
C2	Instrução 2	Instrução 1	
C3			
C4	Instrução 3	Instrução 2	Instrução 1
C5			
C6	Instrução 4	Instrução 3	Instrução 2
C7			
C8	Instrução 5	Instrução 4	Instrução 3
C9			

Figura 5.1: Utilização de pipelines em Glenn.

Medindo o desempenho

MIPS, ou Milhões de Instruções por Segundo, é o termo dado para o número de instruções de código que uma máquina é capaz de realizar a cada segundo. Ela, apesar de ser uma medida, é ineficaz na mensuração do desempenho de processadores diferentes devido as diversas variáveis (tipo de instrução, metódos de execução, entre outros) que podem divergir. Em outras palavras, essa medida é apenas usada em processadores com arquiteturas iguais e obsoleta nos demais.

Outro termo usado na medição de desempenho da máquina é o FLOPS, abreviação de Floating Point Operations per Second, ela mede a quantidade de operações de ponto flutuante (cálculos de processamento) realizados por segundo. Semelhante ao MIPS, ela depende de vários quesitos para medir corretamente.

5.2 Arquitetura RISC e CISC

$$\frac{\text{Tempo}}{\text{Programa}} = \left[\left(\frac{\text{Instruções}}{\text{Programa}}\right) \times \left(\frac{\text{Ciclos}}{\text{Instrução}}\right) \times \left(\frac{\text{Tempo}}{\text{Ciclo}}\right)\right]$$

A fórmula acima é chamada de equação do desempenho de um processador, um modelo que as arquiteturas RISC e CISC, da qual falaremos neste capí-

tulo, seguiram e tentaram melhorar seu desempenho, diminuindo termos à esquerda do sinal de igual.

5.2.1 CISC

A arquitetura CISC (Complex Instruction Set Computer) é uma tecnologia de processador que surgiu durante a década de 70, ela executa um número alto e complexo de instruções por ciclo. Devido ao suporte de muitas instruções ela tem seu tempo de execução demorado. Ela é vista em processadores da intel e AMD.

Essa arquitetura surgiu devido a demanda de diminuição do preço do software, que seria por meio da transferência do preço do software para o do hardware. Na época, o preço do hardware era cada vez mais barato enquanto o do software tendia na direção oposta, a invenção do modelo CISC planejava aliviar essa dissemelhança de custos e simplificar o trabalho dos programadores, que sofriam com compiladores carentes e limitações no tamanho de códigos.

Ele usava da linguagem assembly buscando se igualar a linguagens de alto nível (C ou pascal) por meio de compiladores, poís os projetistas da época pretendiam facilitar a escrita de compiladores, compactar o tamanho dos códigos e facilitar a detecção e correção de erros, baixando os custos do software.

De acordo com alguns (PATTERSON, David A.; SEQUIN Carlo H.), essa arquitetura planejava:

- Reduzir as dificuldades de escrita de compiladores;
- Reduzir o custo global do sistema;
- Reduzir os custos de desenvolvimento de software;
- Reduzir drasticamente o software do sistema;
- Reduzir a diferença semântica entre linguagens de programação e máquina;
- Fazer com que os programas escritos em linguagens de alto nível corressem mais eficientemente;
- Melhorar a compactação do código;
- Facilitar a detecção e correcção de erros.

Para melhorar seu desempenho (tempo por programa), foi feito uma tentativa de reduzir o número de instruções por programa, o que também resultou em uma menor quantidade de memória necessária por programa. Algumas táticas utilizadas para esse fim foram a implementação de variação de tarefas em todos tipos de código e os modos de endereçamento complexo.

5.2.2 RISC

A arquitetura RISC (Reduced Instruction Set Computer) é a arquitetura de processador caracterizada pelo baixo número de instruções, das quais são simples, em que as instruções são capazes de serem executadas em um único ciclo do caminho de dados. Ela tem um número de instruções limitado e faz uso intenso de pipelines. Em geral, essa arquitetura prevalece no mercado moderno, sendo usada em modelos de processadores Apple, Motorola e IBM.

A arquitetura surgiu da ideia de realizar mais rápidas as tarefas mais comuns, o inverso da filosofia que inventou a arquitetura CISC. Para isso, ela repassava a complexidade do hardware para o software, descomplicando os conjuntos de instruções, dado que a memória estava se tornando mais barata e os compiladores cada vez mais eficientes.

Na RISC um conjunto de instruções era capaz de realizar a maioria do trabalho, visto sua filosofia que buscava reduzir o número e a quantidade de instruções por programa para obter maior desempenho. Ela também realizou a implementação do *pipelining*, que permitia a execução de várias instruções paralelamente, mas que apenas era viável quando não encarava diferentes graus de complexidade de instruções. Essa técnica colaborou para uma diminuição drástica no número médio de ciclos por instrução (CPI ou Cycles per instruction) e num aumento na quantidade de instruções por programa, o que foi compensado devido ao próprio *pipelining* e outras implementações, como a extinção dos modos de endereçamento complexos e o aumento do número de registos internos do processador.

Exercícios

1) Processadores RISC e CISC são ainda hoje fonte de discussão na hora de se escolher a configuração adequada de um projeto. Espera-se, normalmente, que processadores RISC:
a) façam em uma instrução o que os processadores CISC fazem em muitas.
b) possuam instruções simples, executadas em um único ciclo
c) realizem instruções mais longas, porém mais ricas em semântica.
d) reduzam o número de instruções de máquina para executar um programa

compilado.
e) sejam construídos em torno de um núcleo CISC.

2) Sobre a arquitetura RISC é correto afirmar o seguinte:
a) Em comparação com a CISC, RISC apresenta uma arquitetura com poucos registradores.
b) O uso de pipeline é uma característica da RISC.
c) Comumente, as instruções RISC consomem vários ciclos de clock.
d) é comum, em uma arquitetura RISC, encontramos instruções de 32, 64, 128, 256 e 512 bytes no mesmo processador.
e) Muitas instruções RISC são executadas pelo microcódigo.

5.3 Taxonomia de Flynn

Devido a demanda de computadores mais potentes e capazes de resolver uma alta gama de instruções complexas, surgiram diversas técnicas e arquiteturas para minimizar o tempo de velocidade dessas máquinas, reduzindo a presença do gárgalo entre memória e processador, exemplo disso foi a introdução da cache e de pipelines, a primeira com função de diminuir a quantidade de acessos a memória e a segunda capaz de executar simultaneamente até 3 instruções á frente.

Além delas, outra otimização adotada foi a arquitetura de processador Superescalar, que usa de paralelismo onde ocorrem a execução de diversas instruções simultaneamente (frequentemente 4), em que as instruções necessitam de unidades de processamento (ULA, UC...) disjuntas para serem executadas independentemente.

Dito isso, podemos proseguir para a Taxonomia de Flynn, uma classificação baseada nos fluxos de controle e de dados na execução de instruções.

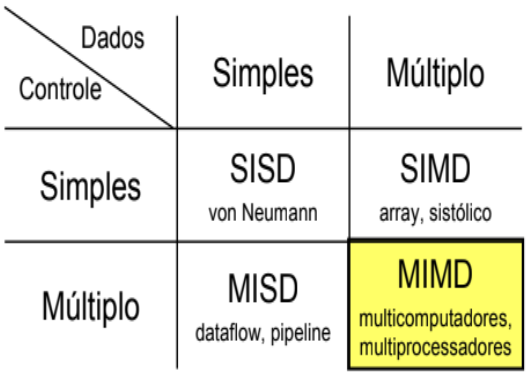


Figura 5.2: Divisão das arquiteturas na taxonomia de Flynn

SISD

SISD (Single Instruction Single Data): Classificação em que um único fluxo de instruções atua sobre um único fluxo de dados.

Figura 5.3: SISD

SIMD

SIMD (Single Instruction Multiple Data): Classificação em que um fluxo único de instruções atua em múltiplos conjuntos de dados.

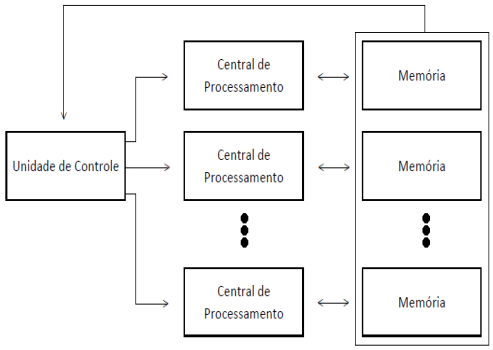


Figura 5.4: SIMD

MISD

MISD (Multiple Instruction Single Data): Classificação em que um fluxo múltiplo de instruções opera em um único conjunto de dados.

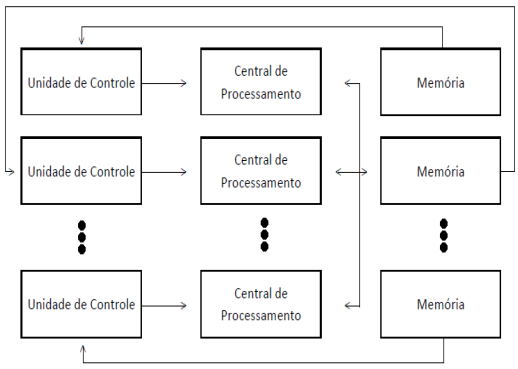


Figura 5.5: MISD

MIMD

MIMD (Multiple Instruction Multiple Data): Classificação em que um fluxo múltiplo de instruções opera em múltiplos conjuntos de dados.

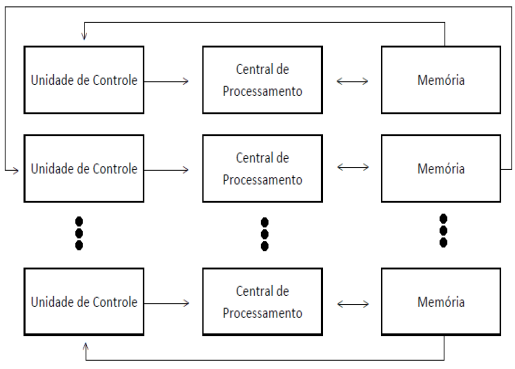


Figura 5.6: MIMD

Dentro delas, a mais relevante, e a qual discutiremos sobre, é a arquitetura MIMD. Ela é um tipo de arquitetura genérica usada pela maioria dos computadores com sistema de multiprocessamento. Nela existem diversas unidades de controle que executam processos semelhantes e compartilham da mesma memória, ou seja, ela executa várias instruções em unidades diferentes que acessam a memória atráves do barramento lógico.

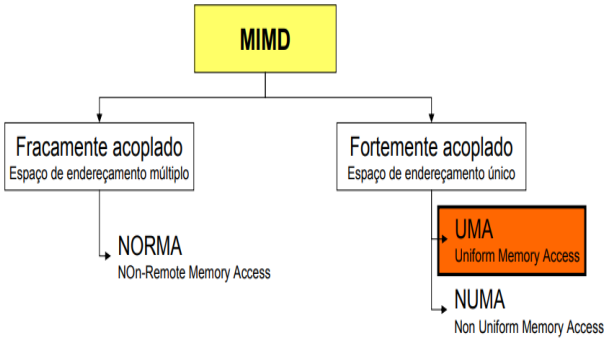


Figura 5.7: Divisão dos tipos de MIMD

Ela possui dois tipos de sistemas, o fortemente e o fracamente acoplado. Um sistema fortemente acoplado tem como característica uma única memória física compartilhada entre diversos processadores com a gerência de dispositivos de E/S atribuida ao sistema operacional. No fracamente acoplado existem dois ou mais sistemas computacionais interligados, mas que funcionam independentes com cada um responsável pela gerência de seus próprios recursos.

A UMA (Uniform Memory Access) é um tipo de classificação de multiprocessador fortemente acoplado que possui tempo de acesso constante independente de qual seja o endereço de memória acessado. Nele cada processador possui sua própria memória cache e são todos idênticos, operando conforme a arquitetura de Von Neumann. Essa arquitetura também possui acesso a memória compartilhado.

5.4 Processador hyperthreading, multiprocessador SMP e processador multicore

Antes de adentrarmos no assunto principal dessa seção devemos fazer menção a programação Multithread, um termo que se refere a divisão de processos, ou linhas de instruções, em duas ou mais tarefas chamadas de threads (linhas) que podem ser executadas simultaneamente e são suportadas pelo próprio sistema operacional. É o uso de threads que dá a ilusão do computador estar executando vários programas ao mesmo tempo sendo que ele apenas realiza um por vez.

Para melhor compreensão podemos aplicar esse conceito na máquina de Glenn. Nela, por exemplo, podemos ter diversas realizações de um mesmo processo ao mesmo tempo, apenas com valores diferentes, com cada um possuindo seu próprio PC.

Abaixo segue uma comparação dos diferentes tipos de arquitetura multi-core, em que o processador possui dois ou mais núcleos no interior de um único chip.

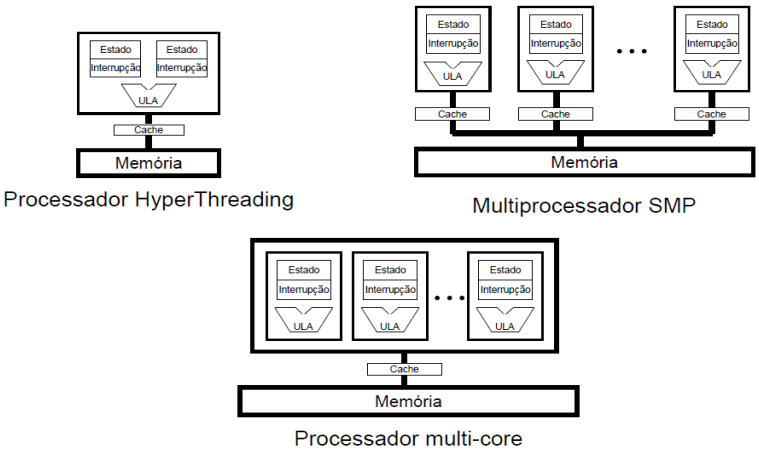


Figura 5.8: Tipos de processador

HyperThreading é o nome dado a arquitetura em que ocorre a simulação de dois processadores lógicos em apenas um físico, isto é, o Sistema Operacional considera que há dois núcleos num processador que tem apenas um.

A arquitetura SMP (Symmetric Multiprocessor) é formada por diversos processadores trabalhando em paralelo e compartilhando uma memória comum a todos. Nela há um Sistema Operacional que possui dever de atribuir as atividades para os vários processadores na rede. Cada unidade possui uma memória cache individual.

A arquitetura Multi-Core possui um único chip integrado a vários núcleos, o que, segundo alguns (CALHEIRO, Gerson; SANTOS, Rafael) resulta na multiplicação total dos recursos de processamento. Ela possui uma única memória cache compartilhada pela rede.

6

Capítulo 6 - Sistemas operacionais

Neste capítulo discutiremos brevemente sobre o Sistema operacional (SO), que é um conjunto de programas que tem como função inicializar e gerenciar os recursos do computador, oferecendo integridade e segurança ao sistema. O SO também provê uma interface para o usuário interagir com a máquina, desprezando a necessidade de conhecer a língua do hardware.

6.1 Conceitos

O sistema operacional possui algumas funções principais, são elas o gerenciamento de processos (hardware, software e endereçamento), o gerenciamento de processador, o gerenciamento de memória, o gerenciamento de arquivos e o gerenciamento de dispositivos (E/S).

A função de gerenciamente de processos se refere a própria gerência de recursos mencionada anteriormente, é ela que trabalha para que não ocorram conflitos entre recursos. Apesar do nome semelhante, a tarefa do gerenciamento de processador é outra distinta, é ele que decide quem usará o processador, por meio de prioridades, e qual será seu tempo de processo, ou timeslice, que é o tempo que ele poderá prosseguir sem interrupções.

O gerênciamento de memória tem papel de armazenar os dados dos processos, que são programas em execução, o que é bastante importante para o funcionamento regular do computador. Por exemplo, o CPU está executando o processo A e, ao acabar seu timeslice, ele se transfere para o processo B, guardando o PC e as informacões do processo A. O mesmo aconteceu quando o timeslice de B acaba, ele retorna para onde A parou, guardando o dados de B. Isso permite que, ao trocar a execução dos processos, seu progresso não seja perdido.

O gerenciamento de arquivos e responsável pela parte de alocação, criação e exclusão de diretórios e arquivos no sistema. E, por ultimo, o gerenciamento de dispositivos de entrada e saida trata da comunicação a qualquer dispositivo ligado ao computador, informando ao usuario o estado do aparelho e quais ações podem ser tomadas.

6.2 Processo e Memória virtual

O processo possui uma estrutura que pode ser dividida em três partes: o contexto de hardware, o contexto de software e o espaço de endereçamento. Abaixo se encontra uma ilustração dessa estrutura.

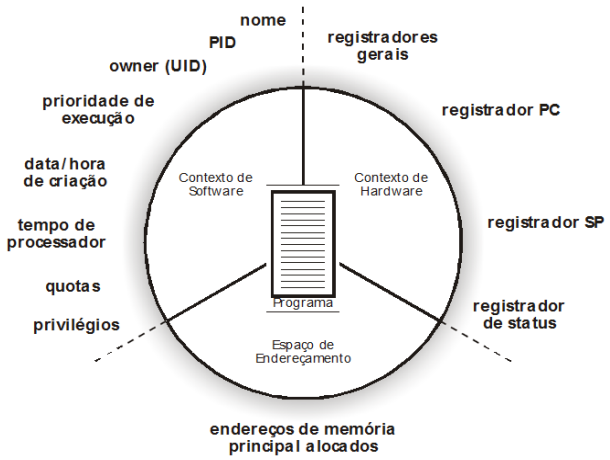


Figura 6.1: Estrutura do processo

O processo também possui estados que representam sua condição atual no processamento. A imagem abaixo ilustra eles.

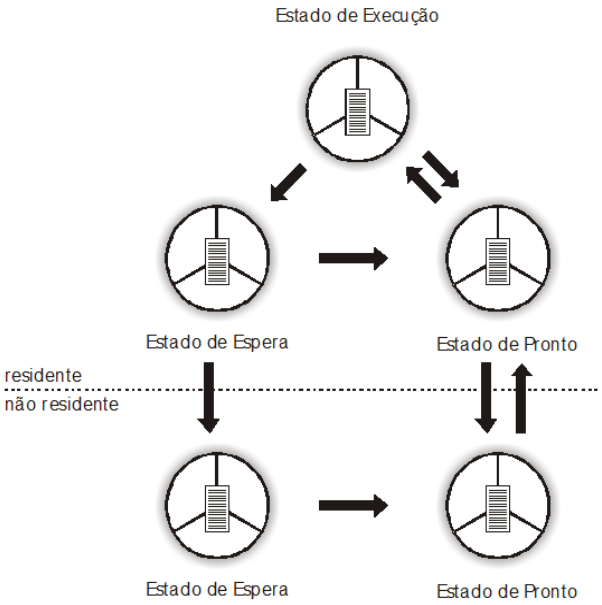


Figura 6.2: Estados do processo

- Estado de execucão: o processo atual a ser executado pelo CPU;
- Estado de espera: aguardando entrada de informação pelo usuário (ociosidade do CPU). A prioridade depende do algoritmo/processo;
- Estado de pronto: o processo está pronto para ser executado pelo CPU.

Além disso, podemos dividir os processos em residente e não-residente. Não-residente é quando ele pode estar pronto, mas nao está na memória enquanto residente é quando, mesmo com apenas uma página, ele se encontra na memória.

Páginas são divisões de N quantidades iguais, isto é, cada página possui a mesma quantidade de elementos. Elas são separadas em página virtual e página real. Virtual é quando ela está sendo usada mesmo que não preenchida/carregada, real é quando está preenchida/carregada.

Os processos podem ser executados em *foreground* ou *background*, ou seja, em primeiro plano ou segundo plano, respectivamente. Processos de *foreground* são visualizados pelo usuário que deve interagir com o processo para avançar-lo, enquanto processos de *background* não necessitam dessa interação e procedem, muitas vezes, sem que o usuário saiba de sua existência.

Memória Virtual, ou arquivo de paginação, é o nome dado a um espaço de memória do disco rígido que atua como uma extensão da memória RAM, funcionando como uma memória auxiliar (especificamente uma cache) em alguns casos.

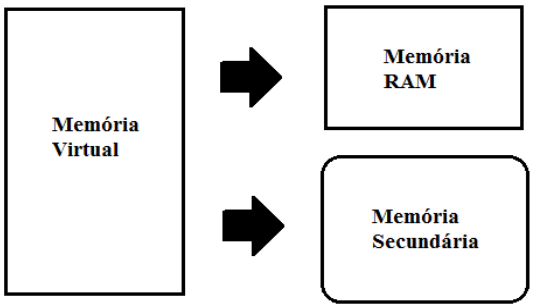


Figura 6.3: Memória Virtual

SOSIM

SOSIM é um software desenvolvido por Luiz Paulo Maia, Mestre em Informática pelo Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro. SOSIM é uma ferramenta que simula os conceitos de multiprogramação, processo e suas mudanças de estado, gerência do processador (escalonamento) e a gerência memória virtual.

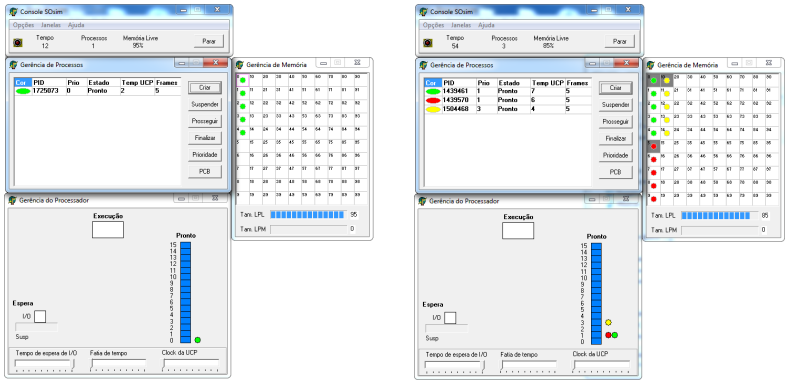
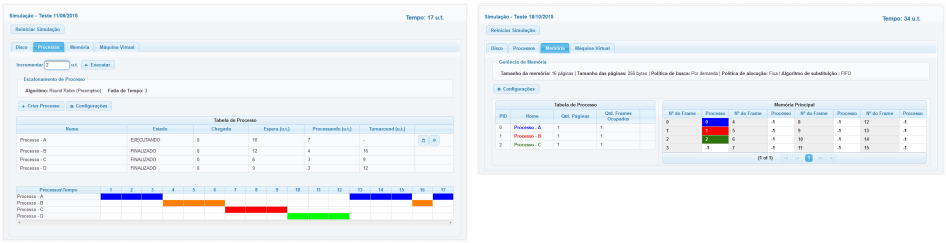


Figura 6.4: SOSIM

SWSO

SWSO é um simulador web de sistemas operacinais desenvolvido por Ramon Almeida Oliveira, Graduado em Análise e Desenvolvimento de Sistemas pelo IFBA campus Salvador. Com o SWSO é possível simular o funcionamento

de mais de dez diferentes algoritmos, entre escalonamento de processo, escalonamento de disco e substituição de páginas na memória. Além disso, ficou muito facilitada a inclusão de novos algoritmos, abrindo assim a possibilidade para a produção de conhecimento por parte do aluno no ambiente.



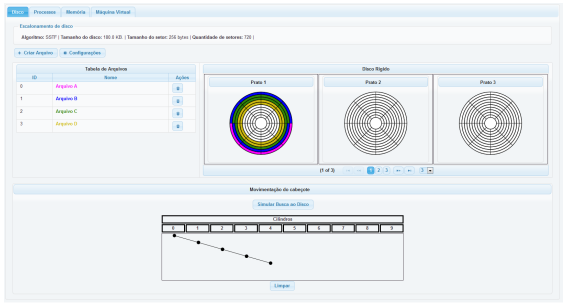


Figura 6.5: SWSO

6.3 Processo de boot

O processo de inicialização, ou processo de boot, do computador é o procedimento em que os componentes da máquina são carregados, nele também é carregado o sistema operacional na memória do computador.

Dentro disso existem várias etapas em que a primeira é a alimentação de energia á máquina, que faz com que o CPU execute uma instrução para que ocorra a leitura da BIOS, que acessa o CMOS para estabelecer uma comunicação com os componentes essenciais da máquina. A BIOS então realiza um auto-teste, chamado de POST (Power On Self Test) para verificar a condição dos hardwares do computador. Finalizada a verificação, a BIOS busca, entre vários, um dispositivo pré-configurado, que pode ser um disco

rígido, pendrive ou outro dispositivo, com que possa iniciar o computador. Caso ainda não encontre, a BIOS apresentará um erro e parará o processo.

Detectado o dispositivo de iniciação, a instrução da BIOS encaminha para a leitura do MBR (Master Boot Record), um setor de inicialização (região de um dispositivo de armazenamento de dados que possui um código a ser carregado pela RAM) responsável por guardar um código de inicialização, que busca uma partição lógica ativa capaz de iniciar o computador.

Caso seja encontrada a partição, é feita a leitura do *Bootloader* (carregador de boot), que carrega e executa o kernel, ou núcleo, um componente principal do sistema operacional que faz a conexão entre hardware e software. O kernel, em seguida, carrega apenas os componentes essenciais do sistema, iniciando-o.

7

Bibliografia

ANTONELLIS, C. J. (2008). Solid state disks and computer forensics. ISSA Journal, pages 36–38.

BROOKSHEAR, J. Glenn. Ciência da Computação: Uma Visão Abrangente. 7. ed.: Bookman, 2004. 512 p.

CORRÊA JÚNIOR, M. A. C. ; QUEIROZ, R. J. G. B. . Forense em Solid State Drive: entendendo os mecanismos internos que podem inviabilizar a recuperação de dados. In: XV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais, 2015, Florianópolis/SC. Anais do XV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais. Porto Alegre - RS: Sociedade Brasileira de Computação - SBC, 2015. p. 524-533.

CRISTO, Fernando de; PREUSS, Evandro; FRANCISCATTO, Roberto. Arquitetura de computadores. UFSM, Santa Maria. 125 p. 2013.

DELGADO, J.; RIBEIRO, C. Arquitetura de Computadores. 2 ed. LTC, 2009.

DITZEL, David R.; PATTERSON, David A., "Retrospective on HLLCA. 25 years of the international symposia on Computer architecture (selected papers)", 1998.

DUNTEMANN, Jeff; PRONK, Ron. "Inside the PowerPC Revolution", Coriolis Group, 1994.

MACHADO, F.M., MAIA, L.P. Arquitetura de Sistemas Operacionais, 4 ed., Ed. LTC, 2007.

MAIA, L. P. (2004). “Página do SOsim”. Disponível na Internet em http://www.training.com.br/sosim.

OLIVEIRA, Ramon. (2015). “SWSO - Simulador Web de Sistemas Operacionais”. Disponível na Internet em http://www.labrasoft.ifba.edu.br/swso-simulador-web-de-sistemas-operacionais.

PATTERSON, D. A. ; HENNESSY, J.L. Organização e projeto de computadores – a interface hardware software. 3. ed. Editora Campus, 2005.

ROCATTO, A. L. J. 2006. Tecnologia de discos: IDE/ATA/SATA. Universidade de Campinas.

SILVA, F. H. 2005. Interfaces de Discos Rígidos. Universidade de Campinas.

STALLINGS, W. Arquitetura e organização de computadores: projeto para o desempenho. 8. ed. Prentice Hall, 2009.

TANENBAUM, A. S. Organização Estruturada de Computadores, quinta edição. Pearson Prentice Hall, 2007.

WILLRICH, Roberto. Introdução à Arquitetura de Computadores;. In: WILLRICH, Roberto. Introdução à Informática. Florianópolis: 2000. cap. 3, p. 46-75. Disponível em: <http://www.inf.ufsc.br/ roberto.willrich/Ensino/INE5602/>. Acesso em: 21 out. 2018.

Índice Remissivo