Unidade 2: Introdução à Organização de Computadores

Prof. Daniel Caetano

Objetivo: Introduzir os conceitos básicos do funcionamento de um computador para possibilitar a compreensão do conceito de variáveis em um software.

Bibliografia STALLINGS, 2003; MURDOCCA e HEURING, 2000; BOENTE, 2006.

INTRODUÇÃO

Quando desejamos entender como usar corretamente uma ferramenta, é preciso entender como ela funciona, como pode ser regulada e o que ela pode fazer por nós. No caso dos computadores, isso não é diferente: para que possamos usá-lo em todo seu potencial, precisamos conhecê-lo com alguma profundidade.

Nesta aula iniciaremos conhecendo os principais elementos de um computador e o funcionamento básico de cada um deles. Também será apresentada a base de numeração "binária", que é a "linguagem do computador". Finalmente é apresentada a memória e seu funcionamento básico, bem como a representação de dados nessa memória.

1. O FUNCIONAMENTO GERAL DO COMPUTADOR

O modelo de funcionamento lógico dos computadores modernos foi proposto pela primeira vez por John von Neumann, um dos projetistas e construtores do ENIAC. Em seu modelo, representado simplificadamente pela Figura 1, o computador é composto por 4 elementos fundamentais: a Unidade Central de Processamento (UCP ou CPU), a Unidade de Memória, a Unidade de Entrada e a Unidade de Saída.

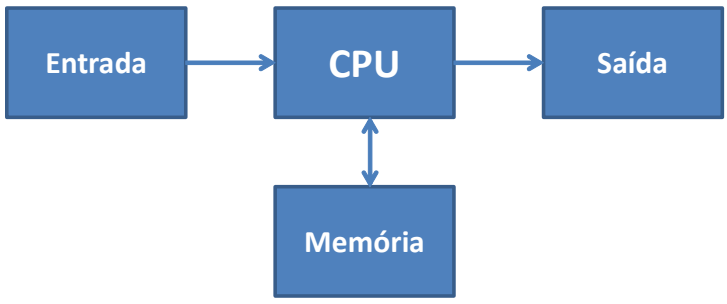

Figura 1: Modelo simplificado baseado no Modelo Von Neumann

A função da CPU é a de coordenar todo o funcionamento do equipamento, além de realizar todos os cálculos necessários. As Unidades de Entrada têm a função de fornecer os dados para processamento; estes dados (**números**) são armazenados na Unidade de

Memória. As Unidades de Saída, por sua vez, são responsáveis por apresentar os resultados dos cálculos ao mundo exterior. Um fato importante é que este foi o primeiro modelo que propôs o armazenamento dos programas (software) na Unidade de Memória, juntamente com os dados em processamento, como ocorre em grande parte dos sistemas computacionais modernos.

Uma sequência possível de funcionamento de um computador é a seguinte:

1. A CPU começa a leitura do programa na memória.
2. Seguindo as especificações do programa, a CPU solicita dados de entrada às unidades de entrada.
3. A CPU realiza as operações de cálculo necessárias e armazena os resultados na memória.
4. Ao término dos cálculos, o programa orienta a CPU a enviar os resultados para alguma das unidades de saída.

1.1. As Partes de uma CPU

Como é possível verificar, a CPU tem umpapel de destaque no processo, aparecendo em praticamente todas as etapas do funcionamento, mas com funções distintas: em algumas situações ela aparece coordenando o processamento e, em outras, realizando cálculos. Por essa razão, von Neumann julgou conveniente representá-la como sendo composta por **duas** partes: a **Unidade de Controle** (UC) e a **Unidade Lógica Aritmética** (ULA).

A Unidade de Controle é responsável por coordenar o funcionamento do computador; é ela a responsável por acessar a memória e executar a sequência lógica de um programa. A Unidade Lógica Aritmética, por sua vez, tem a função de realizar cálculos e operações lógicas, como apoio para o funcionamento da Unidade de Controle.

Em uma analogia simplificada, a UC usando a ULA seria como um ser humano usando uma calculadora. Quem sabe o que e quando precisa ser feito é a UC, mas quem sabe fazer o cálculo é a ULA.

1.2. Dispositivos de Entrada e Saída

Os dispositivos de Entrada são, fundamentalmente, aqueles utilizados para entrada de dados no computador. Os mais clássicos são o teclado e mouse, passando pelas telas de toque... no entanto, estes dispositivos podem ser representados pela mais vasta gama de sensores, como os de temperatura e movimento. Quando usamos um programa, os dispositivos de entrada fornecem ao computador dados para processamento; quando programamos, esses mesmos dispositivos fornecem ao computador uma lógica a ser seguida.

Do ponto de vista do computador, os dispositivos de entrada têm a função de converter as informações fornecidas pelo usuário para uma forma numérica que ele seja capaz de entender e armazenar em sua memória.

Os dispositivos de saída são aqules cuja função é apresentar resultados ao usuário. Esses dispositivos são encontrados nas mais variadas formas, desde os clássicos monitores e impressoras, passando por dispositivos geradores de som, controladores de equipamentos mecânicos, dentre outros.

Do ponto de vista do computador, os dispositivos de saída têm a função de converter as informações numéricas fornecidas pelo computador para uma forma que o ser humano seja capaz de compreender, através do uso de imagens e sons.

> Atenção: *pen drives*, *HDs* e outros dispositivos do gênero são classificados como unidades de memória (armazenamento). Relaxando um pouco o rigor, se considerarmos a troca de informações entre mais de um computador e falarmos em dispositivos de armazenamento removíveis, é possível interpretá-los como dispositivos de entrada ou saída, dependendo da situação de uso.

2. A MEMÓRIA PRINCIPAL

A Unidade de Memória pode ser dividida em dois tipos de memória: a memória permanente e a memória volátil. A memória permanente é aquela em que armazenamos nossas informações para que elas possam ser reutilizadas no futuro. Exemplos de memória permanente são os discos (removíveis ou não) e fitas magnéticas.

A memória volátil, por outro lado, tem a principal função de armazenar as informações utilizadas pelos computadores durante seu processamento: enquanto nosso computador está funcionando, é lá que encontraremos o nosso programa e as informações que por ele são tratadas. Por ser indispensável ao funcionamento do computador, a memória principal é também chamada de **memória principal**.

A memória principal é usada para guardar informações. Como a razão de guardar informações é poder utilizá-las depois, é fundamental que todas as informações armazenadas na memória possam ser, posteriormente, recuperadas. Para que isso seja possível, a memória do computador é organizada como se fosse um grande arquivo de fichas.

Cada uma das fichas é única e possui um nome numérico também único. Esse número que dá nome à ficha é chamado de **endereço de memória**. Como cada ficha pode armazenar um único dado, pode-se dizer que cada dado na memória fica associado a um endereço de memória. Cada uma dessas fichas é chamada de **posição de memória**.

Podemos imaginar, por exemplo, que a memória do computador seja como um enorme arquivo de gavetas numeradas e que cada uma dessas gavetas pode guardar um dado diferente. Se quisermos o dado armazenado na gaveta 2376, basta abrir a gaveta 2376 e pegar esse dado. No computador o processo é o mesmo: se quisermos o valor armazenado na cujo endereço é 2376, basta solicitarmos o dado do endereço de memória 2376.

É comum representarmos a memória como uma tabela, indicando os valores existentes em cada posição de memória:

Endereço	0	1	2	3	4	5	6	7
Valor	10	57	0	255	100	7	10	2

Como já foi dito, o computador só lida com números e, assim, em sua memória apenas números são armazenados. Entretanto, o computador não "vê" os números da mesma maneira que nós. Para entender a forma com que o computador entende os números, é interessante verificar primeiramente a forma como nós entendemos os números.

3. A LINGUAGEM DO COMPUTADOR

Nós, seres humanos, possuímos cinco dedos em cada uma das mãos, totalizando dez dedos nas mãos, como um todo. Isso fez com que construíssemos um sistema numérico bastante peculiar: o sistema decimal.

No sistema decimal, usamos dez elementos para representar os números: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9. Em um número decimal, cada um destes elementos recebe o nome de **dígito**. Cada dígito de um número decimal representa um multiplicador de uma potência de dez. Observe os exemplos:

Número		Milhar	Centena	Dezena	Unidade
27	=			2*10 +	7*1
256	=		2*100 +	5*10 +	6*1
1537	=	1*1000 +	5*100 +	3*10 +	7*1

Numeremos os dígitos da direita para a esquerda, começando em zero, como indicado a seguir, usando o número 1537 como exemplo:

Dígito	3	2	1	0
Número	1	5	3	7

É possível, então, perceber que o número que indica a posição do dígito representa o expoente da potência de dez que o multiplica:

Nome do Dígito	Posição	Multiplicador
Unidade	0	$1 = 10^0$
Dezena	1	$10 = 10^1$
Centena	2	$100 = 10^2$
Milhar	3	$1000 = 10^3$

Assim, de uma forma um pouco menos comum, é possível descrever os números como se segue:

$27 = 2*10^1 + 7*10^0$
$256 = 2*10^2 + 5*10^1 + 6*10^0$
$1537 = 1*10^3 + 5*10^2 + 3*10^1 + 7*10^0$

Observe que existe um claro padrão.

3.1. O Computador e os Números

Como apresentado anteriormente, acredita-se que os humanos trabalhem com números decimais por conta da quantidade de dedos que temos nas mãos; os computadores, entretanto, foram construídos com uma outra característica, que torna a representação numérica mais natural em um outro formato.

Como se sabe, os computadores são construídos por peças eletrônicas como fios e transístores. Apesar de ser possível representar uma ampla gama de valores com um único fio, através de diferentes tensões, os engenheiros consideraram que isso seria muito pouco prático, além de menos confiável, já que podem ocorrer variações significativas de tensão com a mudança da temperatura dos equipamentos.

Assim, foi definido o conceito da ***eletrônica digital***, em que um fio/conexão devem transmitir apenas um de dois valores: ligado (com tensão) ou desligado (sem tensão). Essa decisão tem um impacto bastante relevante na forma com que representamos as informações em um computador: considerando que o fio é a "mão" do computador, ele considera apenas dois dígitos: 0 e 1 - diferentemente de nós, que consideramos os dígitos de 0 a 9!

A informação que pode ser representada por um "fio" - 0 ou 1 - é denominada **bit**, e é a menor unidade de informação de um computador. Se um processador tivesse apenas 1 bit, ele só seria capaz de representar os números 0 e 1. Mas, e se ele tiver 2 bits? A tabela abaixo mostra todas as combinações possíveis:

00 01 10 11

E se ele tiver 3 bits?

000 001 010 011 100 101 110 111

E se ele tiver 4 bits?

0000 0001 0010 0011 0100 0101 0110 0111
1000 1001 1010 1011 1100 1101 1110 1111

Observe que o número de bits representa o número de dígitos binários; além disso, considerando **n** bits, o número de combinações possíveis é dado por 2^n: um computador com 8 bits pode representar até $2^8 = 256$ números e um de 16 bits pode representar até $2^{16} = 65536$ números... e assim por diante.

> Nota: como 256 variações eram suficientes para representar a maior parte das informações necessárias nos primeiros computadores, o conjunto de 8 bits ganhou um nome específico: byte. Assim, um **byte é um conjunto de 8 bits**.

3.2. Conversão de Números Binários para Decimais

Mas que números esse valores representam, em nossa notação decimal?

Existe uma regra de conversão muito simples. Lembremos como representamos o número decimal anterioremente:

Dígito	3	2	1	0
Número	1	5	3	7

$1537 = 1*10^3 + 5*10^2 + 3*10^1 + 7*10^0$

Se fizermos o mesmo com um número binário, por exemplo, o número binário 1101, teremos:

Dígito	3	2	1	0
Número	1	1	0	1

$1101 \text{ binário} = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0$

Observe que os 10^n foram substituídos por 2^n; a razão para isso é que antes estávamos em uma base decimal, agora estamos em uma base binária. Façamos essa conta:

$1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 8 + 4 + 0 + 1 = 13$

Logo: 1101 binário = 13 decimal

Observe que agora temos uma confusão de representação: um número binário poderia ser lido erradamente como um número decimal. Para evitar esse problema, é usual acrescentar a letra "b" após valores binários, para evitar confusão. Em outras palavras, a representação é a seguinte:

Texto	Valor (em decimal)
1101	1101
1101b	13

A tabela a seguir mostra a conversão das 16 combinações de números de 4 bits de binário para decimal:

Binário	Decimal	Binário	Decimal
0000b	0	1000b	8
0001b	1	1001b	9
0010b	2	1010b	10
0011b	3	1011b	11
0100b	4	1100b	12
0101b	5	1101b	13
0110b	6	1110b	14
0111b	7	1111b	15

2.3. Conversão de Números Decimais para Binários

A conversão de números decimais para binários é similar, e é realizada com um processo de sucessivas divisões inteiras por dois, parando quando a divisão valer 0. Os restos das divisões vão compondo o valor em binário, da esquerda para a direita. Por exemplo: vamos transformar o valor 13 em sua representação binária:

13 / 2 = 6 e sobra... 1
6 / 2 = 3 e sobra... 0
3 / 2 = 1 e sobra 1
1 / 2 = 0 e sobra 1

Assim, o valor 13 é representado em binário como 1101b. Tentemos novamente com outro número maior, 118:

118 / 2 = 59 e sobra... 0
59 / 2 = 29 e sobra... 1
29 / 2 = 14 e sobra... 1
14 / 2 = 7 e sobra... 0
7 / 2 = 3 e sobra... 1
3 / 2 = 1 e sobra... 1
1 / 2 = 0 e sobra... 1

Assim, o valor 118 é representado em binário como 1110110b

4. OS DADOS E A MEMÓRIA

Na grande maioria dos computadores modernos, cada posição de memória tem tamanho suficiente para armazenar exatos **8 bits**, ou seja, **1 byte**. Mesmo em computadores de 32, 64 ou 128 bits, as informações são armazenadas em bytes em memória - daí a sua especificação em múltiplos de byte: quilobyte, megabyte, gigabyte, terabyte...

Lembremos que um byte tem 8 bits e pode representar 256 diferentes números. A tabela a seguir representa o valor 4 armazenado na forma de um byte:

Bit	7	6	5	4	3	2	1	0
Valor	0	0	0	0	0	1	0	0

Essa representação, que considera números inteiros sem sinal, armazena valores que representam os números de 0 a 255; neste caso, os 8 dígitos binários são válidos para representar o número.

Entretanto, é comum que se use um destes bits (o bit 7) para representar o sinal: 0 é positivo e 1 é negativo. Assim, o número -4 poderia ser representado, por exemplo, como segue abaixo:

Bit	7- Sinal	6	5	4	3	2	1	0
Valor	1	0	0	0	0	1	0	0

Isso faz com que os números representados variem, simplificadamente, de -127 a +127.

4.1. Os Números Reais

Como é possível observar, os números apresentados até o momento são todos números inteiros. Como representar números reais, ou seja, números "com vírgula"?

Bem, o computador só pode representar números inteiros, mas existe uma maneira de representar números com vírgula com um truque similar ao do "sinal" dos números inteiros. Consideremos que todo número com vírgula pode ser escrito como um número com a vírgula em posição fixa multiplicado por uma potência de 10. Observe os exemplos:

$1,75 = 1,75 * 10^{0}$
$0,0175 = 1,75 * 10^{-2}$
$17,5 = 1,75 * 10^{1}$
$228,0 = 2,28 * 10^{2}$

A idéia é, então, indicar que todo número real, na memória, será indicado como um número em que a vírgula é considerada fixa entre a primeira e a segunda casa da esquerda para a direita (denominado *mantissa*), multiplicada por uma potência de 10, cujo expoente **também** será armazenado no número em questão:

	Expoente			Número (Mantissa)				
Bit	7 - Sinal	6	5	4 - Sinal	3	2	1	0
Valor	1	1	0	0	1	1	0	0

Nesta tabela, o valor representado na região do número é 12 (verifique!). Assim, como a vírgula da mantissa é fixa entre o primeiro e o segundo númeor, o valor da mantissa

é 1,2. O expoente representado, por sua vez, é o número -2 (verifique!), o que significa se tratar do número $1,2 * 10^{-2} = 0,012$.

Essa representação de número em que temos uma mantissa com ponto fixo e um expoente de potência de 10 é chamada de **representação em ponto flutuante**, porque é a potência de 10 que indica onde realmente está a vírgula (ou seja, a posição da vírgula é variável... ou flutuante).

4.2. Números Maiores

Como pode ser observado, a representação de números inteiros e de números reais é limitada, quando se usa apenas 8 bits. Por essa razão, os números costumam ser interpretados em grupos de dois e quatro bytes, denominados ***word*** (palavra) e ***dword*** (palavra dupla). Isso permite a representação de números bem maiores, como indicados abaixo em dois bytes, o número inteiro 1537 e o real 0,01876 (verifique!):

Bit	15- Sinal	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Valor	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1

	Expoente				Mantissa											
Bit	15- Sinal	14	13	12	11- Sinal	10	9	8	7	6	5	4	3	2	1	0
Valor	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0	0

4.3. As Letras

Até o momento vimos como armazenar números de diferentes tipos na memória. Mas como armazenar letras? Bem, este foi um problema que surgiu nos primórdios da computação e, por esta razão, existe uma solução padrão, que é a chamada Tabela ASCII (ASCII significa *American Standard for Computer Information Interchange*). Esta tabela relaciona cada valor numérico de um byte a cada um dos códigos visuais usados por nós na atividade da escrita. A tabela de conversão é apresentada na página seguinte (fonte: Wikipédia).

Observem, porém, que nem todos os caracteres são definidos por essa tabela: em especial, os caracteres acentuados estão faltando. Mas não são apenas estes: também não estão presentes os caracteres japoneses, chineses, russos... dentre tantos outros.

Por essa razão, atualmente existem diversas outras "tabelas de código de caracteres" ou "páginas de código de caracteres" (do inglês *codepage*), que extendem a tabela abaixo indicando os símbolos faltantes aos códigos livres (não especificados pela tabela ASCII). Entretanto, com a grande troca de arquivos entre pessoas de países diferentes, isso começou a causar alguma confusão. Foi assim que surgiram então os códigos Unicode, que são versões alternativas e universais à tabela ASCII. O padrão UTF (Unicode Transformation Format) define várias tabelas, sendo as mais conhecidas e usadas as tabelas UTF-8 e UTF16.

Binário	Decimal	Hexa	Glifo
0010 0000	32	20	
0010 0001	33	21	!
0010 0010	34	22	"
0010 0011	35	23	#
0010 0100	36	24	$
0010 0101	37	25	%
0010 0110	38	26	&
0010 0111	39	27	'
0010 1000	40	28	(
0010 1001	41	29	)
0010 1010	42	2A	*
0010 1011	43	2B	+
0010 1100	44	2C	,
0010 1101	45	2D	-
0010 1110	46	2E	.
0010 1111	47	2F	/
0011 0000	48	30	0
0011 0001	49	31	1
0011 0010	50	32	2
0011 0011	51	33	3
0011 0100	52	34	4
0011 0101	53	35	5
0011 0110	54	36	6
0011 0111	55	37	7
0011 1000	56	38	8
0011 1001	57	39	9
0011 1010	58	3A	:
0011 1011	59	3B	;
0011 1100	60	3C	<
0011 1101	61	3D	=
0011 1110	62	3E	>
0011 1111	63	3F	?

Binário	Decimal	Hexa	Glifo
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C
0100 0100	68	44	D
0100 0101	69	45	E
0100 0110	70	46	F
0100 0111	71	47	G
0100 1000	72	48	H
0100 1001	73	49	I
0100 1010	74	4A	J
0100 1011	75	4B	K
0100 1100	76	4C	L
0100 1101	77	4D	M
0100 1110	78	4E	N
0100 1111	79	4F	O
0101 0000	80	50	P
0101 0001	81	51	Q
0101 0010	82	52	R
0101 0011	83	53	S
0101 0100	84	54	T
0101 0101	85	55	U
0101 0110	86	56	V
0101 0111	87	57	W
0101 1000	88	58	X
0101 1001	89	59	Y
0101 1010	90	5A	Z
0101 1011	91	5B	[
0101 1100	92	5C	\
0101 1101	93	5D	]
0101 1110	94	5E	^
0101 1111	95	5F	_

Binário	Decimal	Hexa	Glifo
0110 0000	96	60	`
0110 0001	97	61	a
0110 0010	98	62	b
0110 0011	99	63	c
0110 0100	100	64	d
0110 0101	101	65	e
0110 0110	102	66	f
0110 0111	103	67	g
0110 1000	104	68	h
0110 1001	105	69	i
0110 1010	106	6A	j
0110 1011	107	6B	k
0110 1100	108	6C	l
0110 1101	109	6D	m
0110 1110	110	6E	n
0110 1111	111	6F	o
0111 0000	112	70	p
0111 0001	113	71	q
0111 0010	114	72	r
0111 0011	115	73	s
0111 0100	116	74	t
0111 0101	117	75	u
0111 0110	118	76	v
0111 0111	119	77	w
0111 1000	120	78	x
0111 1001	121	79	y
0111 1010	122	7A	z
0111 1011	123	7B	{
0111 1100	124	7C	|
0111 1101	125	7D	}
0111 1110	126	7E	~

5. BIBLIOGRAFIA

STALLINGS, W. **Arquitetura e organização de computadores**. 5ed. São Paulo: Ed. Pearson Prentice Hall, 2003.

MURDOCCA, M. J; HEURING, V.P. **Introdução à Arquitetura de Computadores**. S.I.: Ed. Campus, 2000.

BOENTE, A. **Programação Web Sem Mistérios**. Editora Brasport, 2006.