

Arquitetura e Organização de Computadores – 5cop090



Conceito Básicos e Evolução do Computador

Objetivos do Capítulo

- Explicar as funções gerais e a estrutura de um computador digital.
- Apresentar uma visão geral da evolução da tecnologia dos computadores desde os primeiros computadores digitais até os últimos microprocessadores.
- Apresentar uma visão geral da arquitetura x86.
- Definir sistemas embarcados e listar alguns dos requisitos e das restrições que vários sistemas embarcados podem encontrar.

Organização e arquitetura

- **Arquitetura de computador** refere-se aos atributos de um sistema visíveis a um programador. Ex. conjunto de instruções, o número de bits para representar vários tipos de dados, mecanismos de E/S e técnicas para endereçamento de memória.
- **Organização de computador** refere-se às unidades operacionais e suas interconexões que percebam as especificações de arquitetura. Ex. detalhes do hardware transparentes ao programador (sinais de controle, tecnologia de memória utilizada, interface entre computador e periféricos).
- Historicamente, e ainda hoje, a distinção entre arquitetura e organização tem sido importante.
- Os diferentes modelos na família têm diferentes características de preço e desempenho.

Estrutura e função

- Em cada nível, o projetista está interessado em:
 - **Estrutura:** o modo como os componentes são inter-relacionados.
 - **Função:** a operação individual de cada componente como parte da estrutura.
- Em termos de descrição, temos duas escolhas:
 1. começar de baixo e subir até uma descrição completa, ou
 2. começar com uma visão de cima e decompor o sistema em suas subpartes.

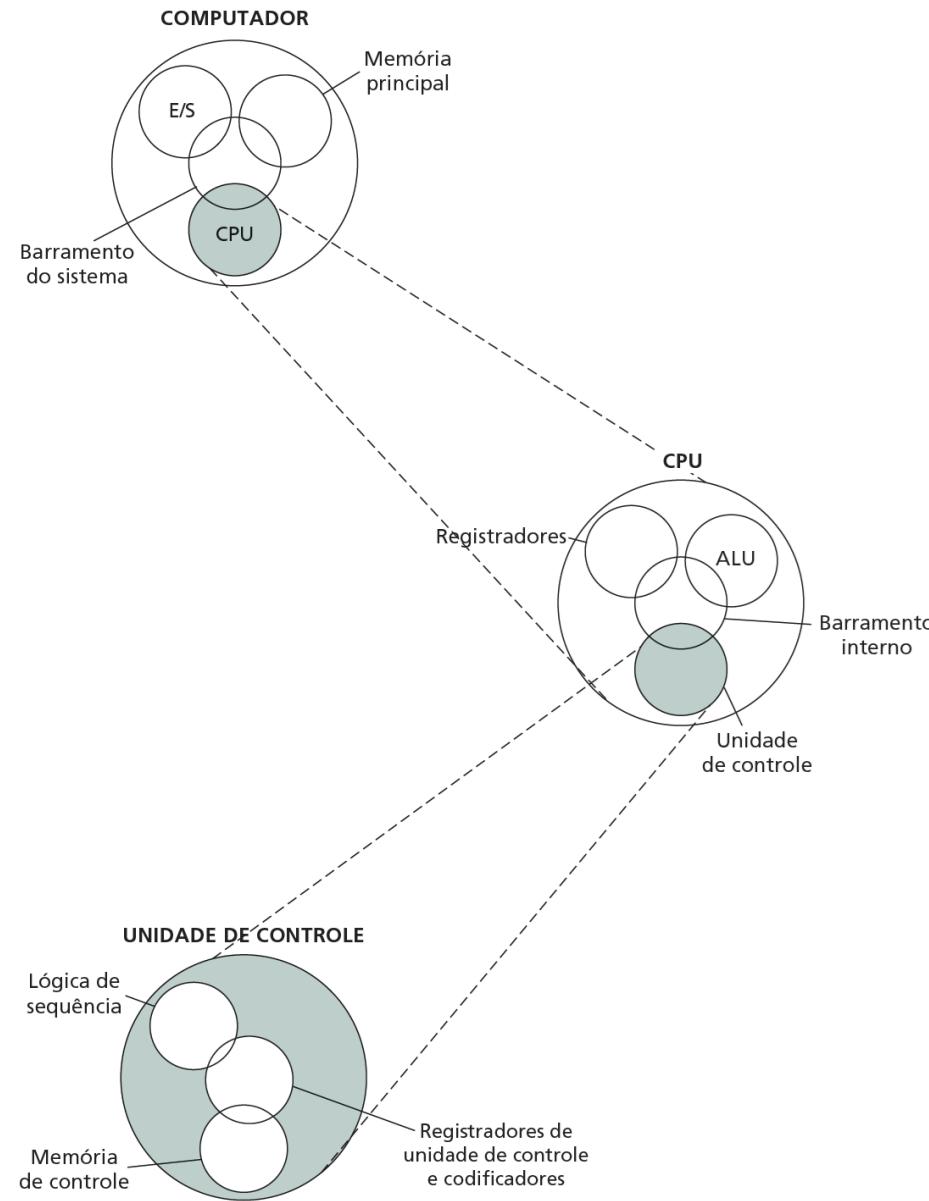
Função

- Em termos gerais, há somente quatro funções básicas que podem ser apresentadas pelo computador:
 1. Processamento de dados
 2. Armazenamento de dados
 3. Movimentação de dados
 4. Controle

Estrutura

- A figura a seguir fornece uma visão hierárquica de uma estrutura interna de um computador de processador único tradicional.
- Há quatro componentes estruturais principais:
 1. **Unidade central de processamento (CPU — do inglês, *Central Processing Unit*).**
 2. **Memória principal.**
 3. **E/S.**
 4. **Sistema de interconexão.**

Estrutura



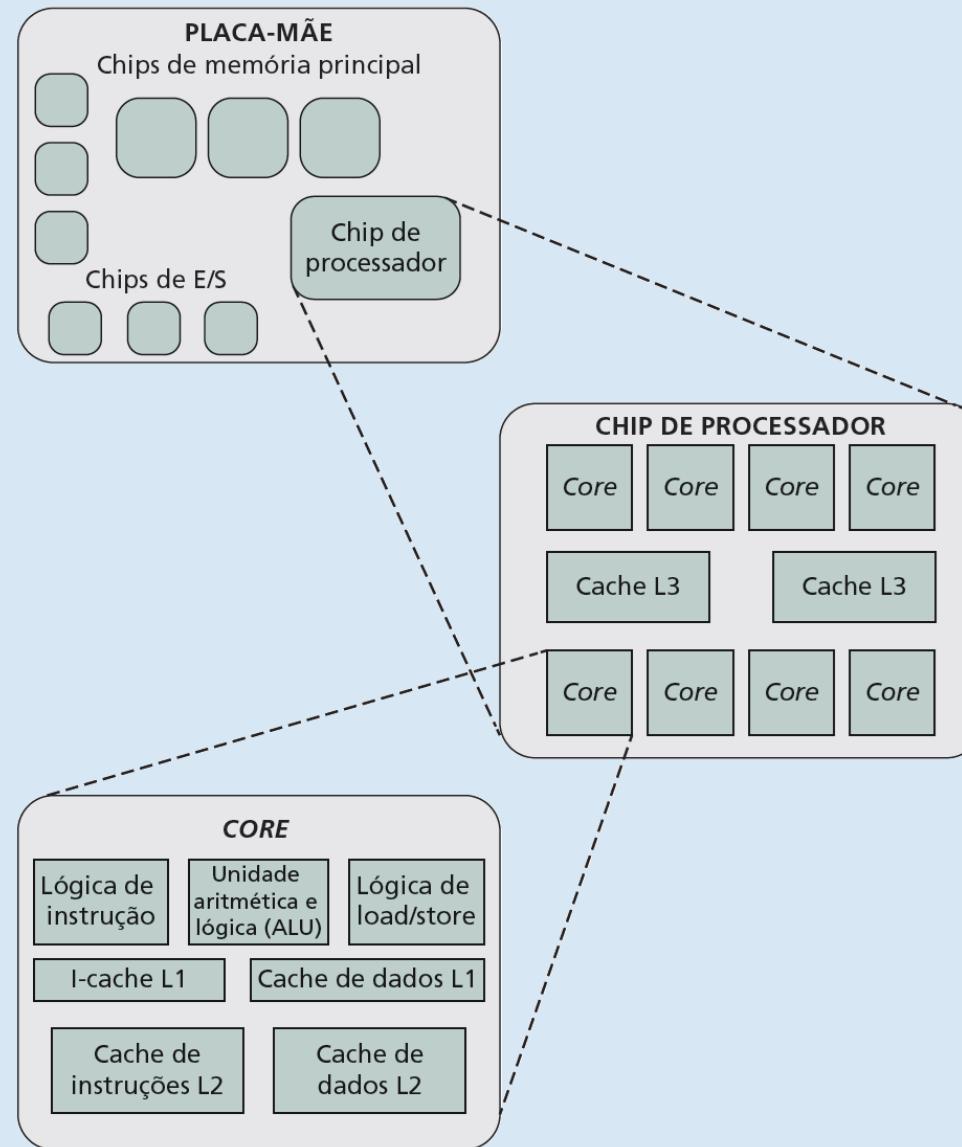
Estrutura

- O mais complexo componente é a CPU.
- Seus principais componentes estruturais são os seguintes:
 1. **Unidade de controle.**
 2. **Unidade lógica e aritmética (ALU — do inglês, *Arithmetic and Logic Unit*).**
 3. **Registradores.**
 4. **Interconexão da CPU.**

Estrutura

- Quando os processadores todos residem em um único chip, o termo *computador multicore* é usado.
- Cada unidade de processamento é chamada de *core*.
- Outra característica proeminente de computadores contemporâneos é o uso de múltiplas camadas de memória, chamada de *memória cache*, entre o processador e a memória principal.
- A figura a seguir é uma visão simplificada dos componentes principais de um computador multicore típico.

Estrutura



Estrutura

- Uma **placa de circuito impresso** (PCB — do inglês, *Printed Circuit Board*) é uma placa rígida e plana que mantém e interconecta chips e outros componentes eletrônicos.
- A placa de circuito impresso principal em um computador é chamada de placa de sistema ou **placa-mãe**.
- Um **chip** é um pedaço único de material semicondutor, em geral de silício, no qual os circuitos eletrônicos e portas lógicas são fabricados.
- O produto resultante é referido como um **circuito integrado**.

Estrutura

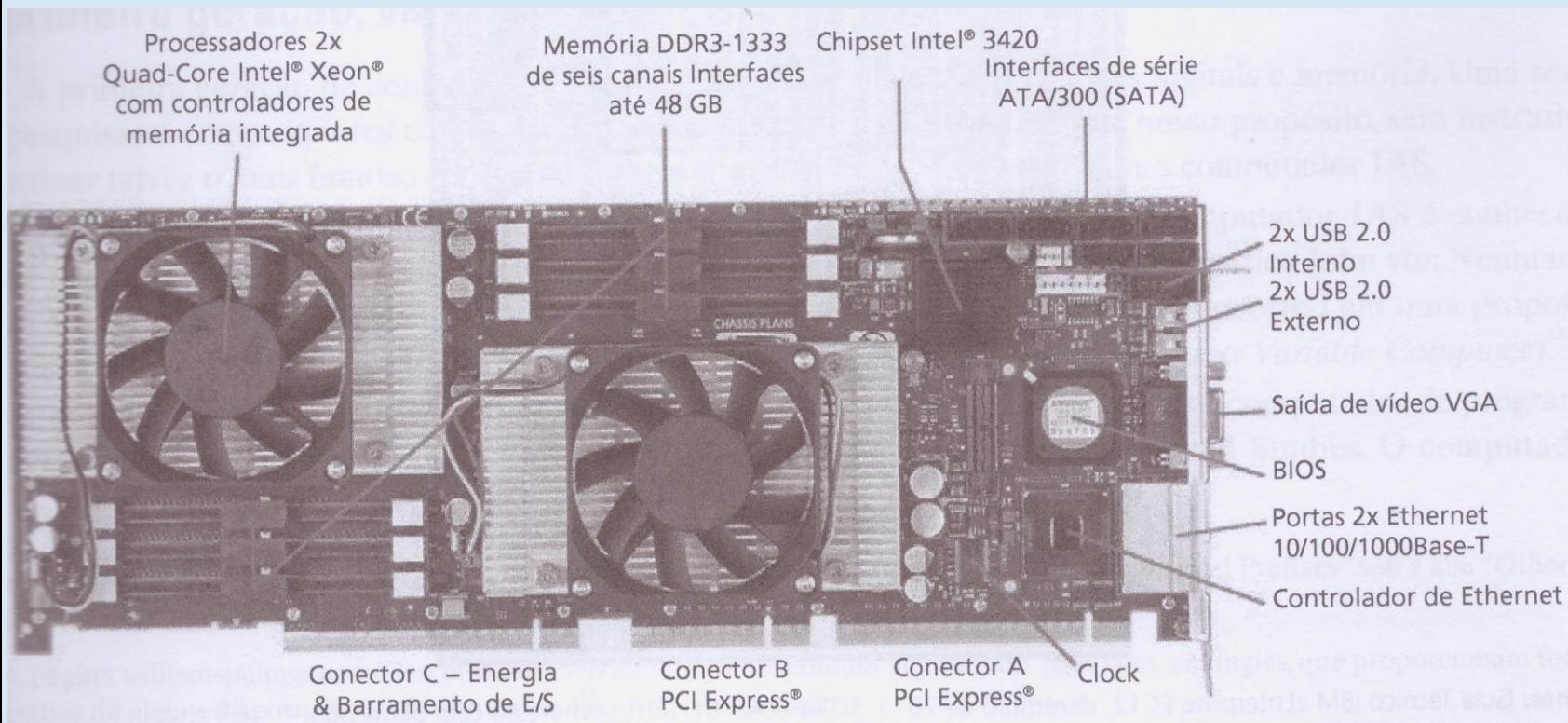
- Em linhas gerais, os elementos funcionais de um core são:
- **Lógica de instrução:** inclui as tarefas envolvidas em buscar instruções, e decodificar cada instrução a fim de determinar a operação de instrução e os locais de memória dos operandos.
- **Unidade lógica e aritmética (ALU):** executa a operação especificada por uma instrução.
- **Lógica de load/store:** gerencia a transferência de dados para e de uma memória principal através da cache.

Estrutura

- Em linhas gerais, os elementos funcionais de um core são:
- **Lógica de instrução:** inclui as tarefas envolvidas em buscar instruções, e decodificar cada instrução a fim de determinar a operação de instrução e os locais de memória dos operandos.
- **Unidade lógica e aritmética (ALU):** executa a operação especificada por uma instrução.
- **Lógica de load/store:** gerencia a transferência de dados para e de uma memória principal através da cache.

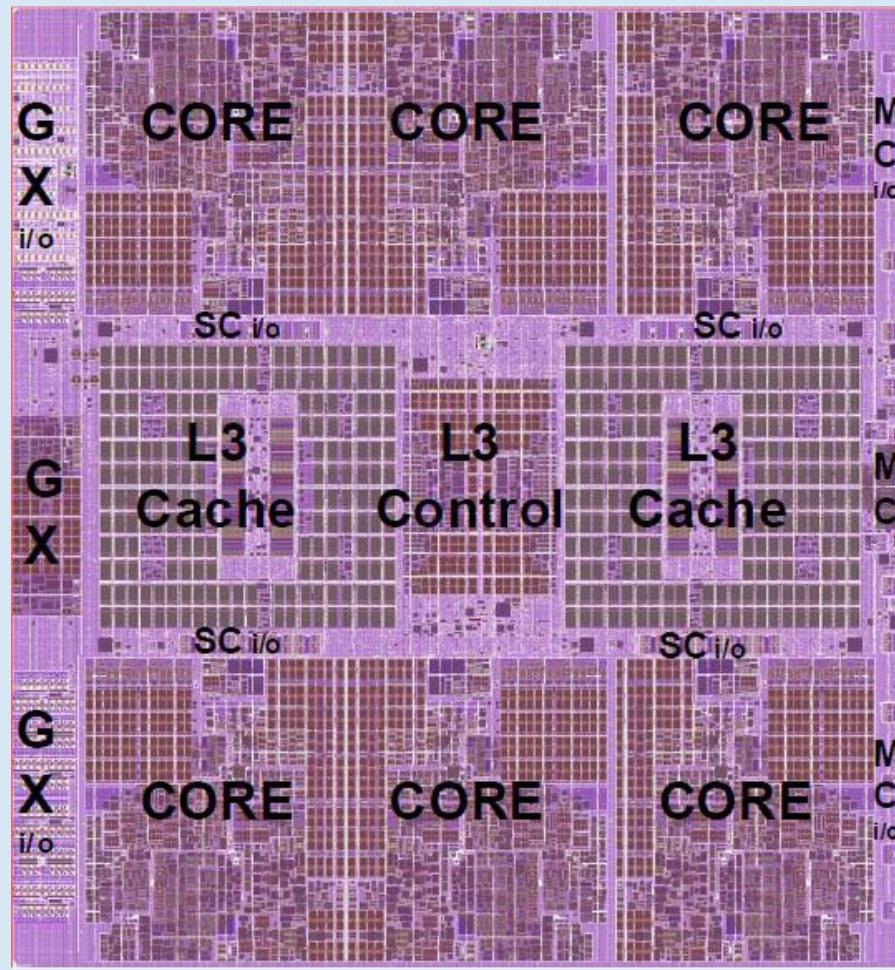
Estrutura

- Placa-mãe com dois processadores Intel Quad-Core Xeon



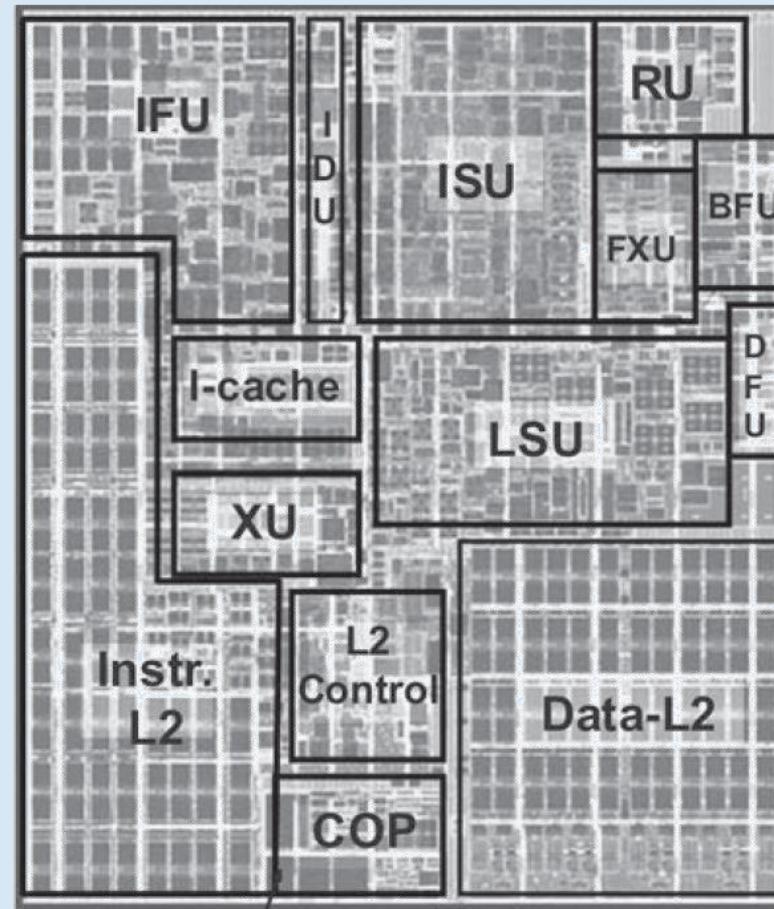
Estrutura

- Diagrama do chip da unidade de processador (UP) zEnterprise EC12



Estrutura

- Layout do core do zEnterprise EC12:



Estrutura

As principais subáreas dentro dessa área do core são as seguintes:

- **ISU** (unidade de sequência de instrução – **Instruction Set Unit**): determina a sequência na qual as instruções são executadas no que é referido como arquitetura superescalar.
- **IFU** (unidade de busca de instrução – **Instruction Fetch Unit**): lógica para buscar instruções.
- **IDU** (unidade de decodificação de instrução – **Instruction Decode Unit**): a IDU é alimentada por buffers IFU e é responsável por analisar e decodificar todos os opcodes de z/Arquitetura).
- **LSU** (unidade de load/store – **Load-Store Unit**): a LSU contém uma cache de dados de 96 kB L1, e gerencia o tráfego de dados entre a cache de dados L2 e as unidades de execução funcionais. É responsável por lidar com todos os tipos de acesso de operandos de todas as extensões, modos e formatos. Como definido na z/Arquitetura.

Estrutura

As principais subáreas dentro dessa área do core são as seguintes:

- **XU** (unidade de tradução – **Translation Unit**): essa unidade traduz os endereços lógicos a partir de instruções nos endereços físicos na memória principal. A XU também contém o **TLB** (Translation Lookaside Buffer) usado para incrementar o acesso da memória.
- **FXU** (unidade de ponto fixo – **Fixed Point Unit**): a FXU executa as operações aritméticas de ponto fixo).
- **BFU** (unidade de ponto flutuante binário – **Binary Floating-point Unit**): a BFU lida com todas as operações de ponto flutuante binário e hexadecimal, bem como com operações de multiplicação de ponto fixo.

Estrutura

As principais subáreas dentro dessa área do core são as seguintes:

- **DFU** (unidade de ponto flutuante decimal – **Decimal Floating-point Unit**): a DFU lida tanto com as operações de ponto fixo como as de ponto flutuante sobre os números que são armazenados como dígitos decimais.
- **RU** (unidade de recuperação – **Recovery Unit**): a RU mantém a cópia do estado completo do sistema que inclui todos os registradores, coleta sinais de falha de hardware e gerencia as ações de recuperação de hardware.
- **COP** (coprocessador dedicado – **Dedicated Co-Processor**): o COP é responsável pela compressão de dados e funções de criptografia para cada core.
- **I-cache**: esta é uma cache de instrução 64 kB L1 que permite que a IFU pré-busque instruções antes que sejam necessárias.

Estrutura

As principais subáreas dentro dessa área do core são as seguintes:

- **Controle L2:** esta é a lógica de controle que gerencia o tráfego através de duas caches L2.
- **Dados-L2:** trata-se de uma cache de dados 1 MB L2 para todo o tráfego de memória diferente das instruções.
- **Instr-L2:** é uma cache de instrução 1 MB L2.

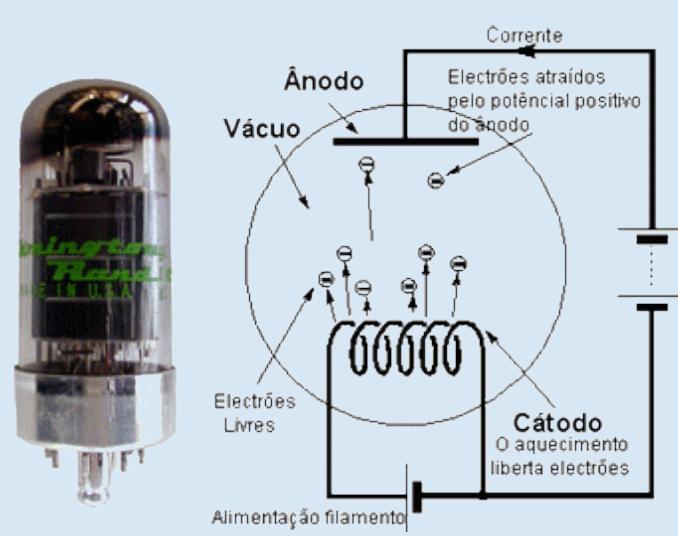
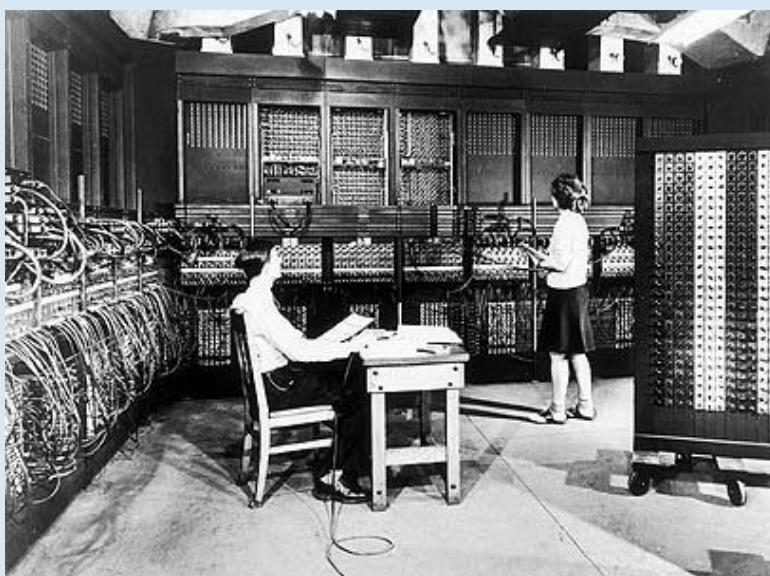
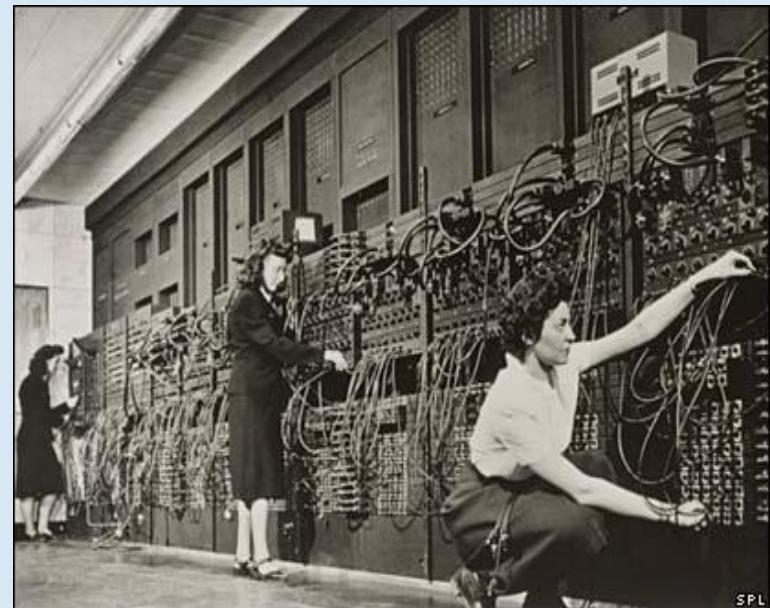
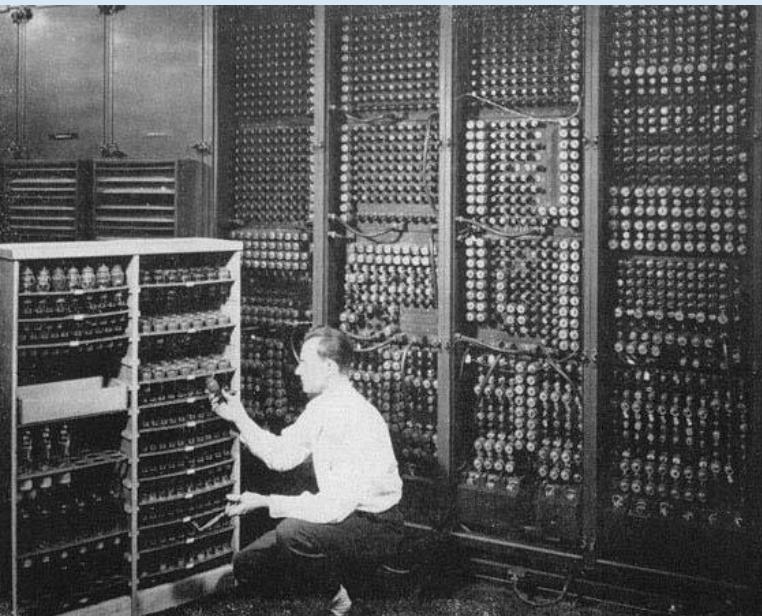
Primeira Geração - Válvulas

ENIAC (Electronic Numerical Integrator And Computer) – histórico e detalhes

- Primeiro computador digital eletrônico de uso geral do mundo.
- 30 toneladas, ocupava uma área de 457,2 m² de superfície e mais de 18000 válvulas, consumo em operação de 140 kilowatts de potência .
- 5000 adições por segundo (máquina decimal com 20 acumuladores).
- 1 anel de 10 válvulas representava cada dígito decimal.
- Programação manual por meio de chaves e conexão e desconexão de cabos.
- John Mauchly e John Eckert (Universidade da Pensilvânia).
- Tabelas de trajetória para armas (II Guerra Mundial - BRL).
- Iniciou em 1943.
- Terminou em 1946.
 - Muito tarde para o esforço de guerra.
- Cálculos para determinação da viabilidade da bomba de hidrogênio.
- Usado até 1955 pelo BRL (*Ballistics Research Laboratory*) quando foi desmontado.

Primeira Geração - Válvulas

ENIAC - Fotos



Primeira Geração - Válvulas

ENIAC - Vídeo

Primeira Geração - Válvulas

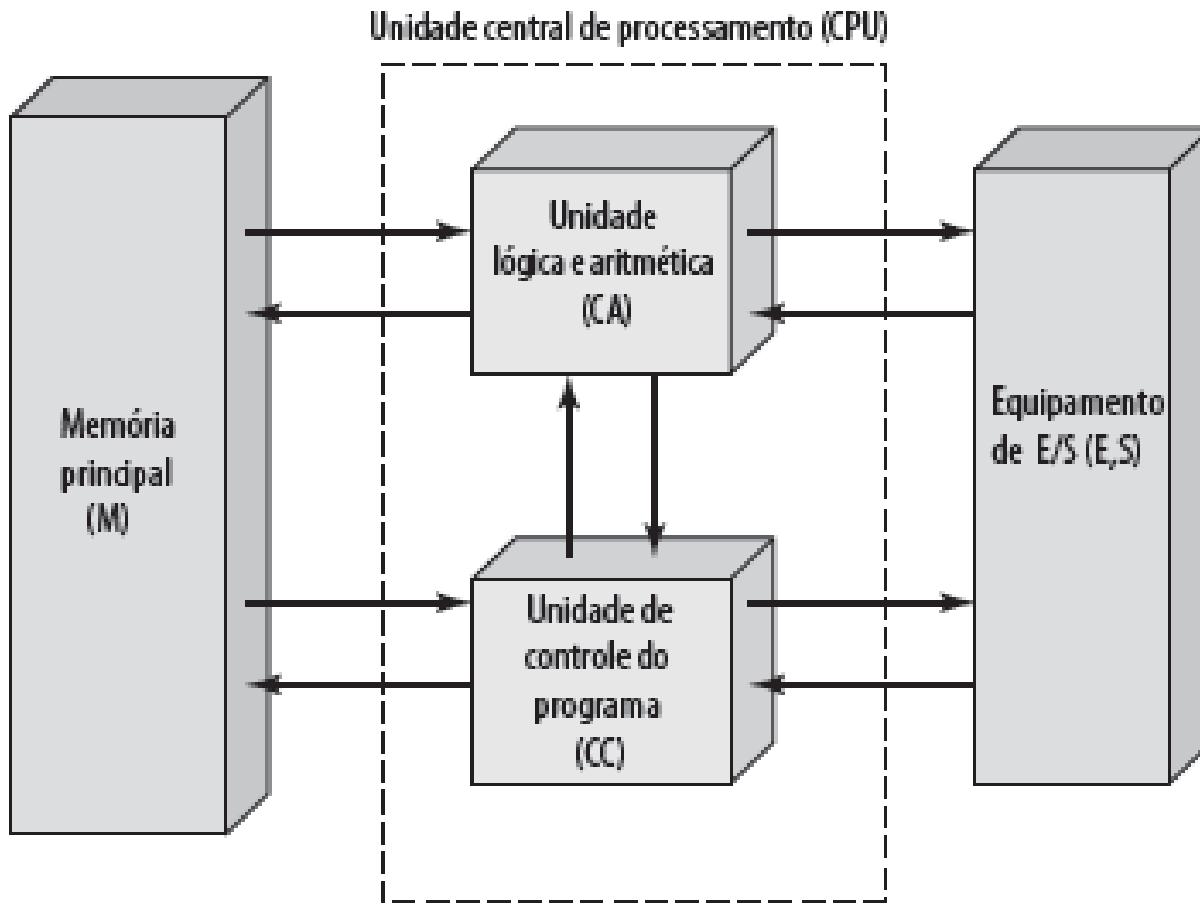
Von Neumann/Turing

- John von Neumann/Alan Turing
- Conceito de programa armazenado.
- Princeton Institute for Advanced Studies - IAS
- Estrutura geral do IAS (concluído em 1952)
 - Memória principal armazenando instruções e dados.
 - ALU operando sobre dados binários.
 - Unidade de controle interpretando e executando instruções da memória.
 - Equipamento de entrada e saída (E/S) operado por unidade de controle.



Primeira Geração - Válvulas

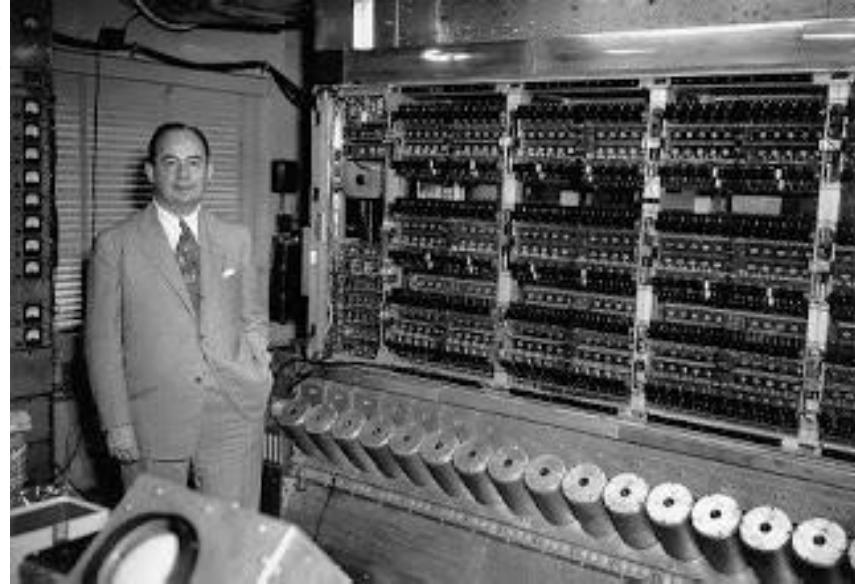
Estrutura da máquina de Von Neumann



Primeira Geração - Válvulas

IAS - detalhes

- 1000 “palavras” de 40 bits.
 - Número binário.
 - 2 instruções de 20 bits.
- 21 instruções.
- Conjunto de registradores (armazenamento em CPU).
 - Registrador de buffer de memória (**MBR** – Memory Buffer Register): contém uma palavra a ser armazenada na memória ou enviada à E/S, ou é usada pra receber uma palavra da memória ou de uma unidade de E/S.
 - Registrador de endereço de memória (**MAR** – Memory Adress Register): especifica o endereço na memória da palavra a ser escrita ou lida no MBR).



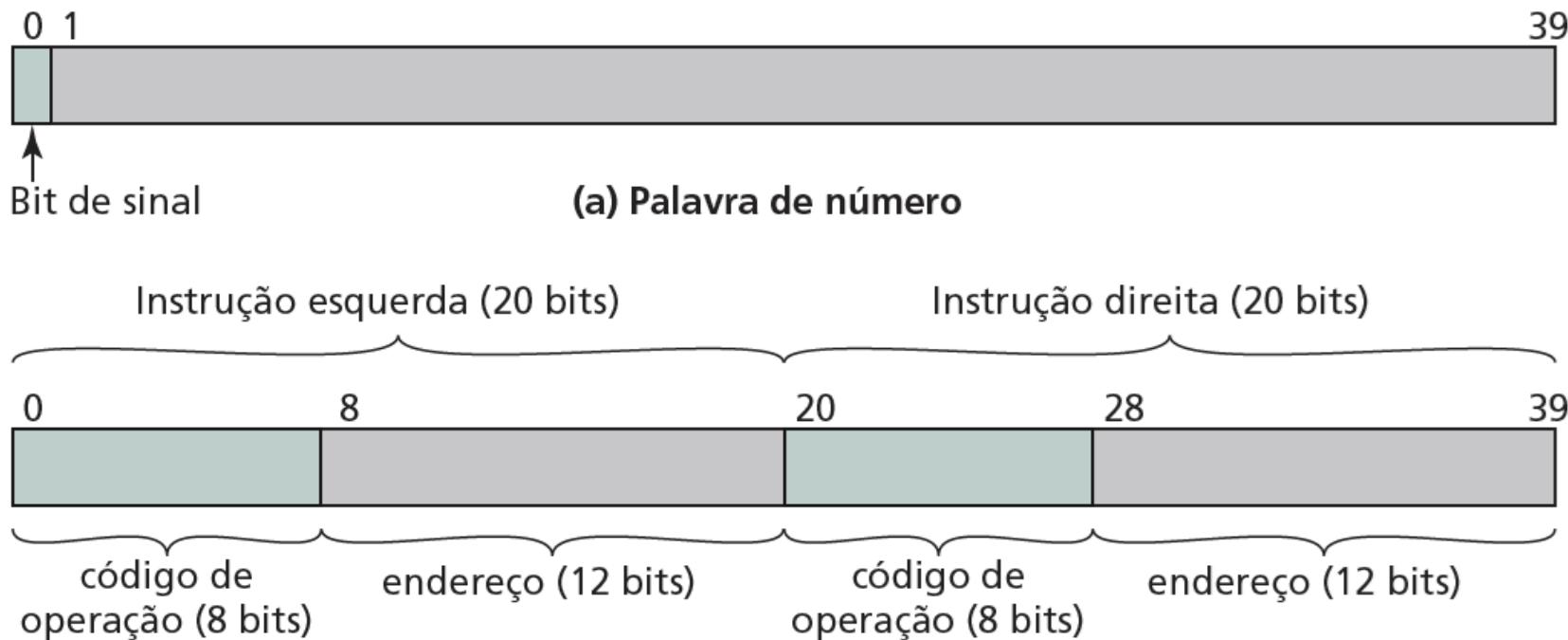
Primeira Geração - Válvulas

IAS - detalhes

- **Registrador de instrução (IR – Instruction Register)**: contém o opcode de 8 bits da instrução que está sendo executada.
- **Registrador de buffer de instrução (IBR – Instruction Buffer Register)**: empregado para manter temporariamente a próxima instrução a ser executada.
- **Contador de programa (PC - Program Counter)**: contém o endereço do próximo par de instruções a ser apanhado da memória.
- **Acumulador (AC) e quociente multiplicador (MQ – Multiplier quotient)**: empregado para manter temporariamente operando de resultados de operações da ALU. Ex. o resultado de multiplicar dois números de 40 bits é um número de 80 bits; os 40 bits mais significativos são armazenados no AC e os menos significativos no MQ.

Primeira Geração - Válvulas

IAS – detalhes – Formato de memória



Primeira Geração - Válvulas

IAS – detalhes – Agrupamento das instruções

- **Transferência de dados:** movem dados entre memória e registradores da ALU ou entre dois registradores da ALU.
- **Desvio incondicional:** utilizadas para facilitar operações repetitivas.
- **Desvio condicional:** o desvio é dependente de uma condição, permitindo assim pontos de decisão.
- **Aritméticas:** operações realizadas pela ALU.

Primeira Geração - Válvulas

IAS – detalhes – Agrupamento das instruções

Instruções de transferência de dados

opcode	mnemônico	significado
00001010	LOAD MQ	$AC \leftarrow MQ$
00001001	LOAD MQ,M(X)	$MQ \leftarrow \text{mem}(X)$
00100001	STOR M(X)	$\text{mem}(X) \leftarrow AC$
00000001	LOAD M(X)	$AC \leftarrow \text{mem}(X)$
00000010	LOAD - M(X)	$AC \leftarrow -\text{mem}(X)$
00000011	LOAD M(X)	$AC \leftarrow \text{abs}(\text{mem}(X))$
00000100	LOAD - M(X)	$AC \leftarrow -\text{abs}(\text{mem}(X))$

Primeira Geração - Válvulas

IAS – detalhes – Agrupamento das instruções

Instruções de desvio incondicional

opcode	mnemônico	significado
00001101	JUMP M(X,0:19)	próx. instr.: metade esq. de mem(X)
00001110	JUMP M(X,20:39)	próx. instr.: metade dir. de mem(X)

Primeira Geração - Válvulas

IAS – detalhes – Agrupamento das instruções

Instruções de desvio condicional

opcode	mnemônico	significado
00001111	JUMP +M(X, 0:19)	se AC ≥ 0 , próx. instr.: metade esq. de mem(X)
00010000	JUMP +M(X, 20:39)	se AC ≥ 0 , próx. instr.: metade dir. de mem(X)

Primeira Geração - Válvulas

IAS – detalhes – Agrupamento das instruções

Instruções aritiméticas

opcode	mnemônico	significado
00000101	ADD M(X)	$AC \leftarrow AC + mem(X)$
00000111	ADD M(X)	$AC \leftarrow AC + mem(X) $
00000110	SUB M(X)	$AC \leftarrow AC - mem(X)$
00001000	SUB M(X)	$AC \leftarrow AC - mem(X) $
00001011	MUL M(X)	$AC:MQ \leftarrow MQ * mem(X)$
00001100	DIV M(X)	$MQ:AC \leftarrow AC / mem(X)$
00010100	LSH	$AC \leftarrow AC * 2$ (shift esq.)
00010101	RSH	$AC \leftarrow AC / 2$ (shift dir.)

Primeira Geração - Válvulas

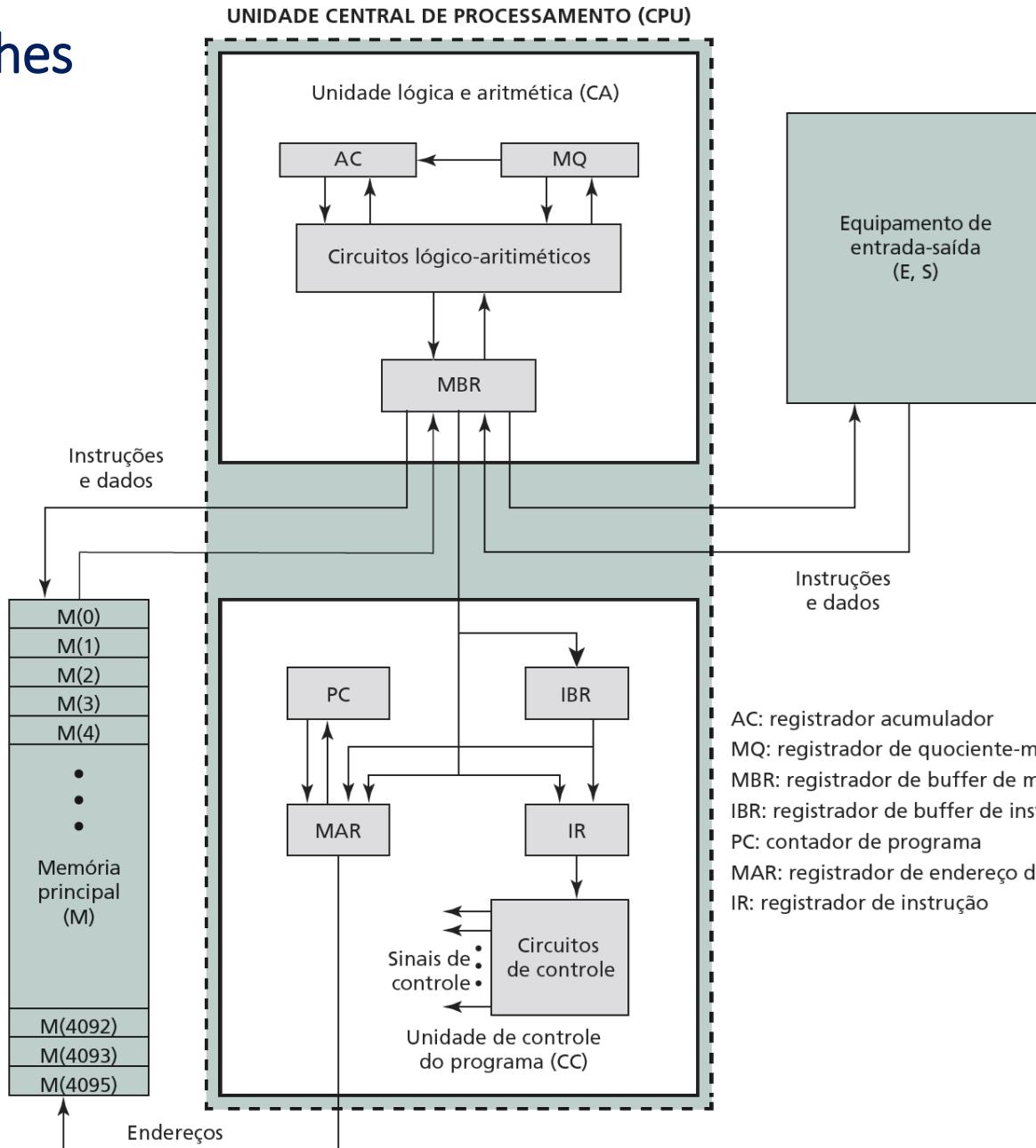
IAS – detalhes – Agrupamento das instruções

Instruções de alteração de endereço

opcode	mnemônico	significado
00010010	STOR M(X,8:19)	substitui o campo de endereço à esq. de mem(X) pelos 12 bits mais à dir. de AC
00010011	STOR M(X,28:39)	subst. o campo de endereço à dir. de mem(X) pelos 12 bits mais à esq. de AC

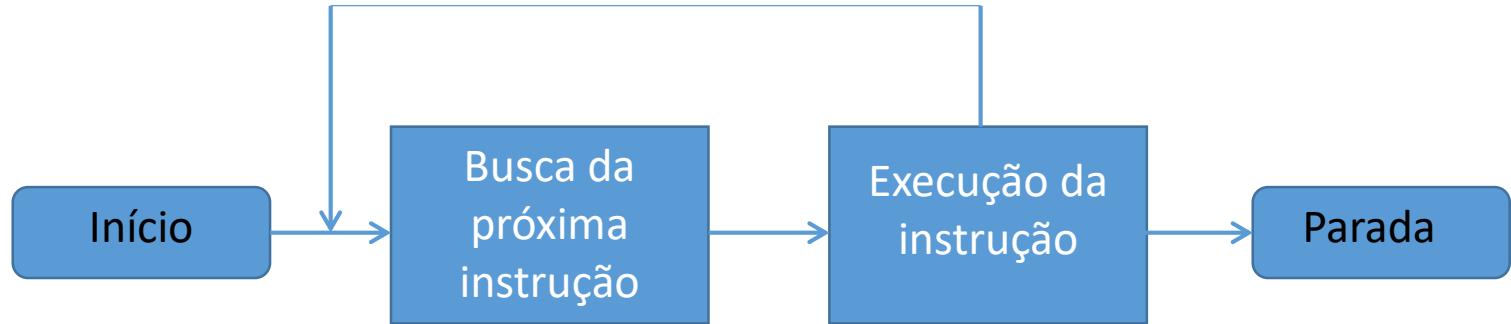
Primeira Geração - Válvulas

IAS – detalhes



Primeira Geração - Válvulas

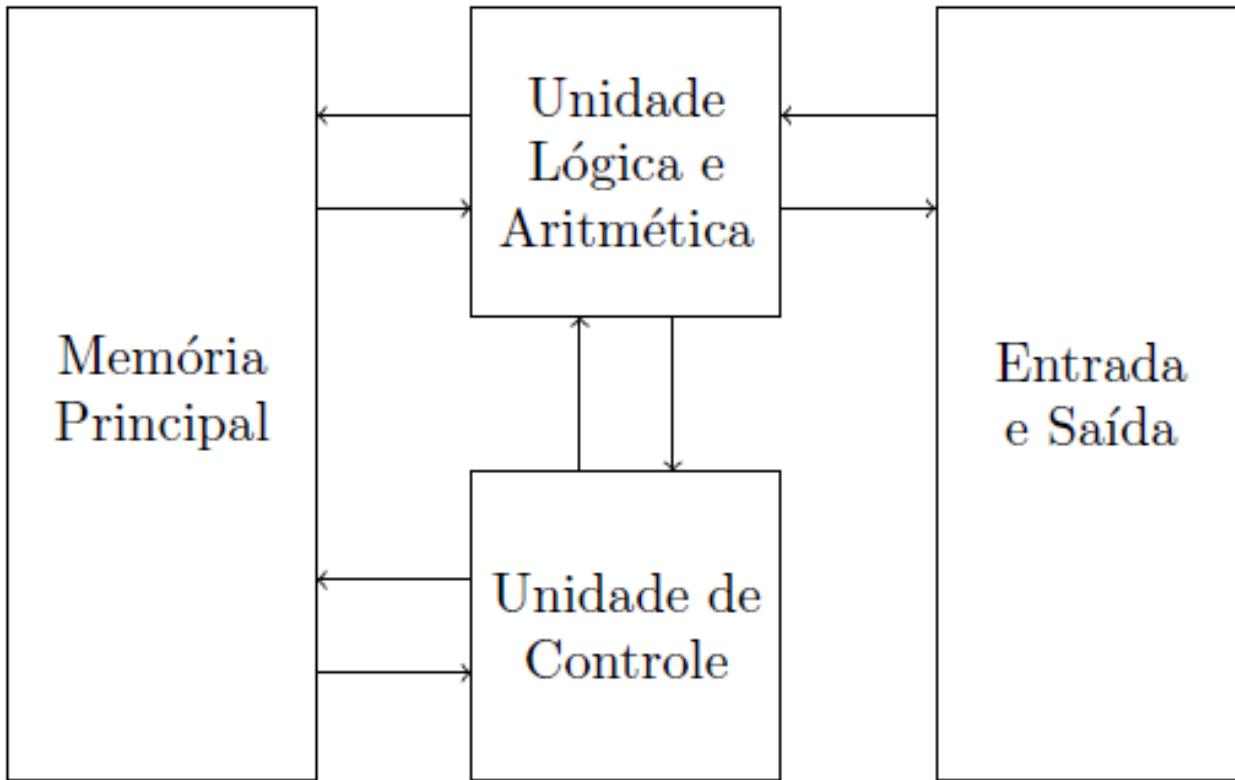
IAS – detalhes – Ciclo de vida de uma instrução



- **Busca:**
 - Busca da instrução
 - Atualiza PC
 - Decodificação
- **Execução:**
 - Busca operandos (se necessário)
 - Executa
 - Armazena resultado (se necessário)

Primeira Geração - Válvulas

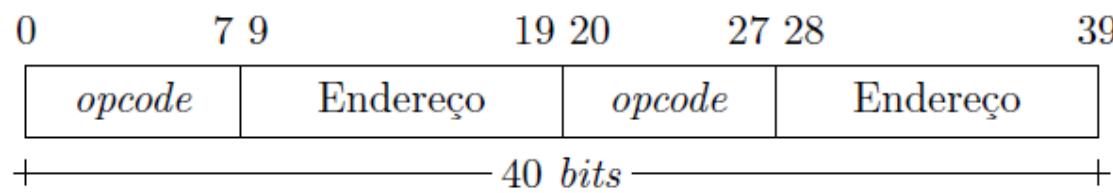
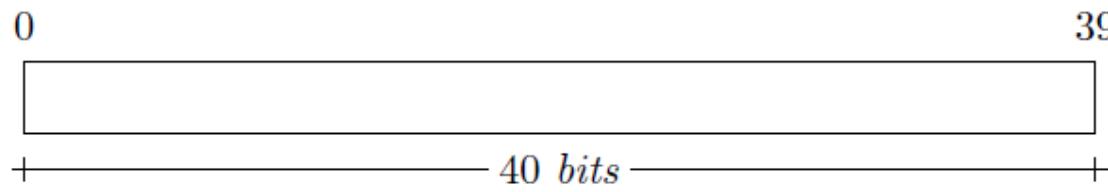
IAS – detalhes – Organização



Primeira Geração - Válvulas

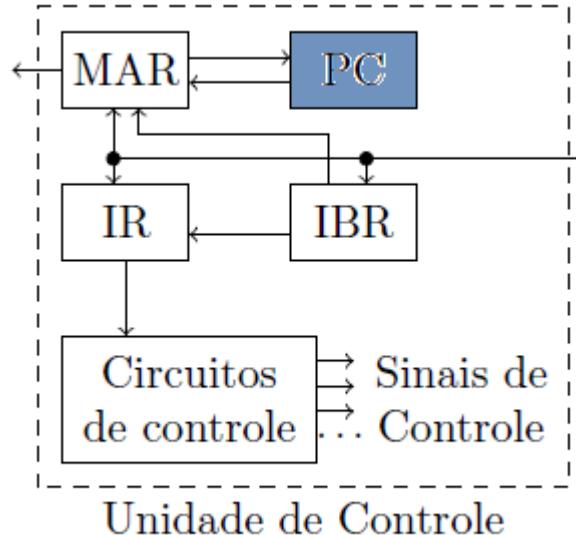
IAS – detalhes – Organização da memória

	40 bits				
0	01	06	90	50	67
1	00	02	6A	01	25
2	01	36	AA	04	11
...
1022	FF	0A	FA	0A	1F
1023	20	1A	F9	10	AB



Primeira Geração - Válvulas

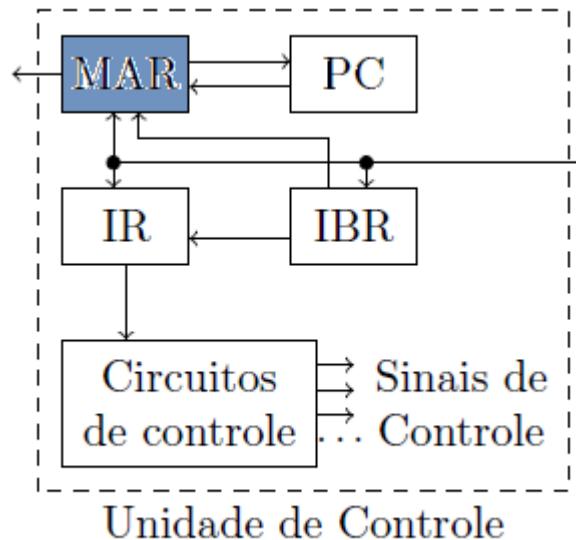
IAS – detalhes – Organização da Unidade de Controle (UC)



PC: o *Program Counter*, ou contador do programa, armazena um valor que representa o endereço da memória que possui o próximo par de instruções a serem executadas. No início, quando o computador é ligado, o conteúdo deste registrador é zerado para que a execução de instruções se inicie a partir do endereço zero da memória.

Primeira Geração - Válvulas

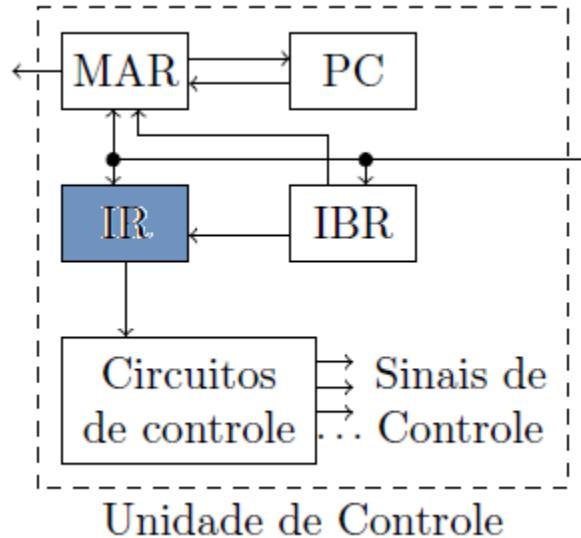
IAS – detalhes – Organização da Unidade de Controle (UC)



MAR: o *Memory Address Register*, ou registrador de endereço da memória, armazena um valor que representa um endereço de uma palavra da memória. Este endereço será lido pela memória durante a operação de leitura ou escrita de dados.

Primeira Geração - Válvulas

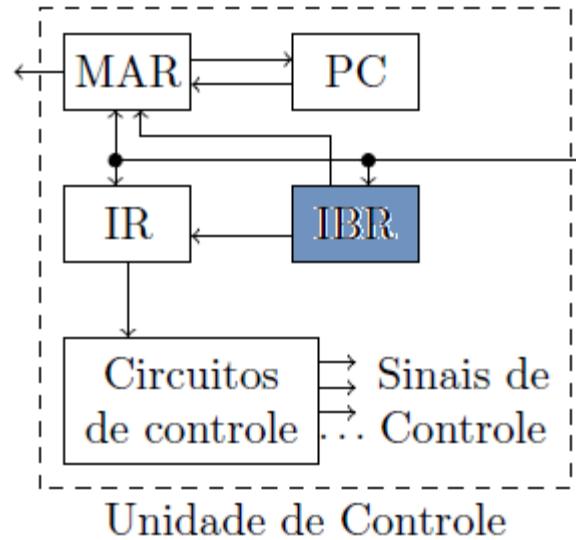
IAS – detalhes – Organização da Unidade de Controle (UC)



IR: o *Instruction Register*, ou registrador de instrução, armazena a instrução que está sendo executada no momento. O circuito de controle da unidade de controle lê e interpreta os bits deste registrador e envia sinais de controle para o resto do computador para coordenar a execução da instrução.

Primeira Geração - Válvulas

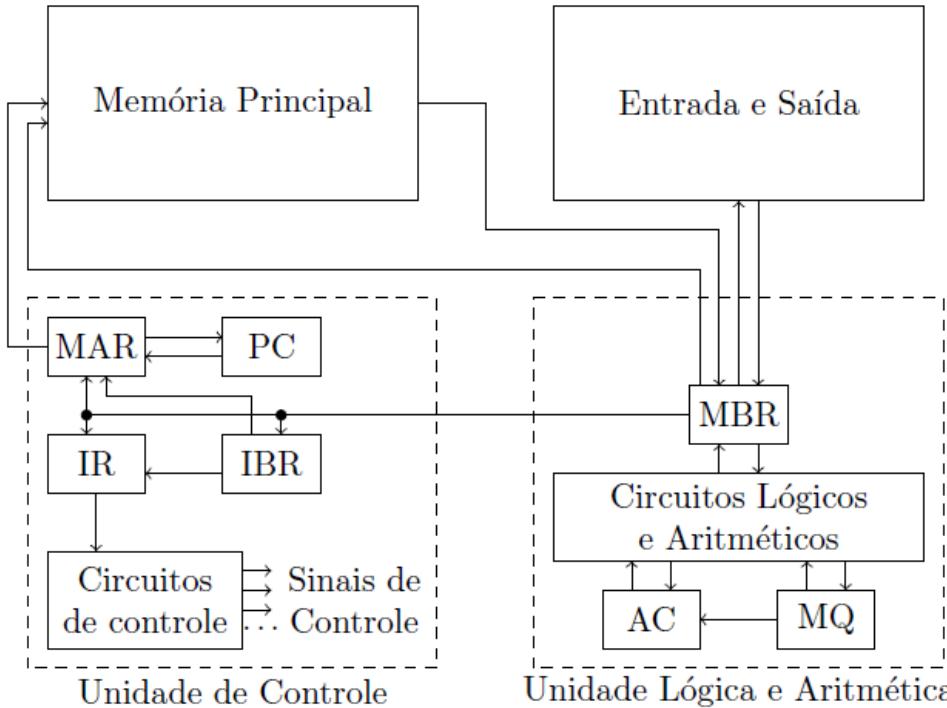
IAS – detalhes – Organização da Unidade de Controle (UC)



IBR: o *Instruction Buffer Register* serve para armazenar temporariamente uma instrução. O IAS busca instruções da memória em pares. Dessa forma, quando o IAS busca um par de instruções, a primeira instrução é armazenada diretamente em IR e a segunda em IBR. Ao término da execução da primeira instrução (em IR), o computador move a segunda instrução (armazenada em IBR) para IR e a executa.

Primeira Geração - Válvulas

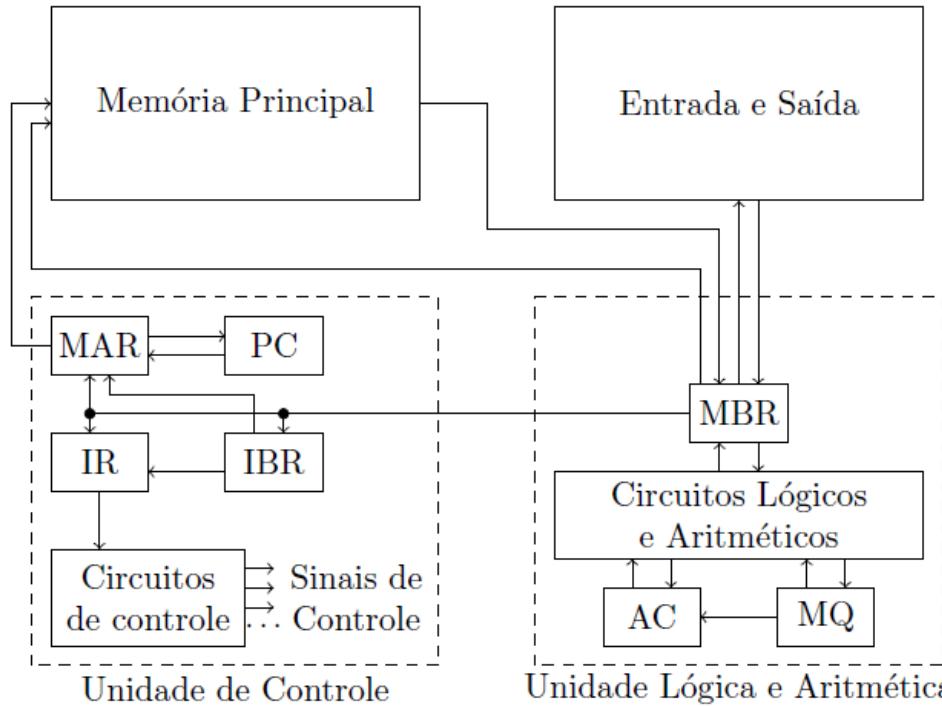
IAS – detalhes – Unidade Lógica e Aritmética (ULA)



MBR: o *Memory Buffer Register*, ou registrador temporário da memória, é um registrador utilizado para armazenar temporariamente os dados que foram lidos da memória ou dados que serão escritos na memória. Para escrever um dado na memória, o computador deve colocar o dado no registrador MBR, o endereço da palavra na qual o dado deve ser armazenado no registrador MAR e, por fim, enviar sinais de controle para a memória realizar a operação de escrita. Assim sendo, os registradores MAR e MBR, juntamente com os sinais de controle enviados pela unidade de controle, formam a interface da memória com o restante do computador.

Primeira Geração - Válvulas

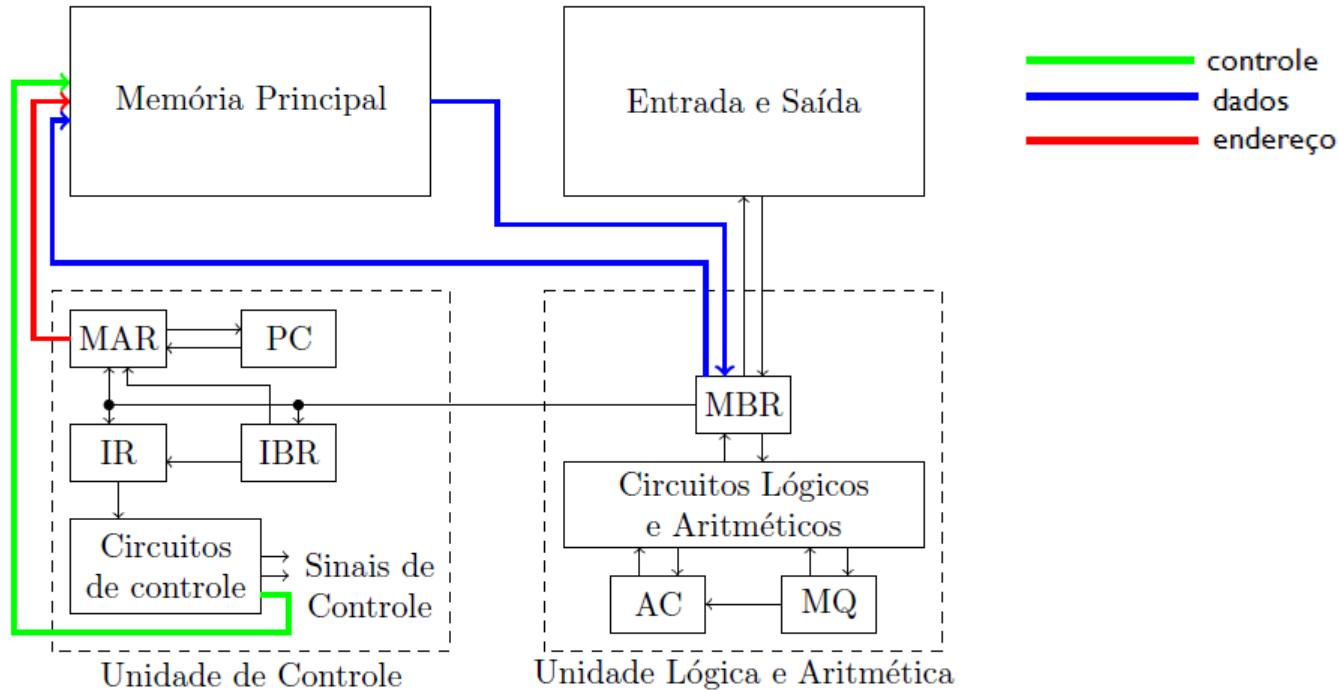
IAS – detalhes – Unidade Lógica e Aritmética (ULA)



AC e MQ: o *Accumulator*, ou acumulador, e o *Multiplier Quotient*, ou quociente de multiplicação, são registradores temporários utilizados para armazenar operandos e resultados de operações lógicas e aritméticas. Por exemplo, a instrução que realiza a soma de dois números (ADD) soma o valor armazenado no registrador AC com um valor armazenado na memória e grava o resultado da operação no registrador AC.

Primeira Geração - Válvulas

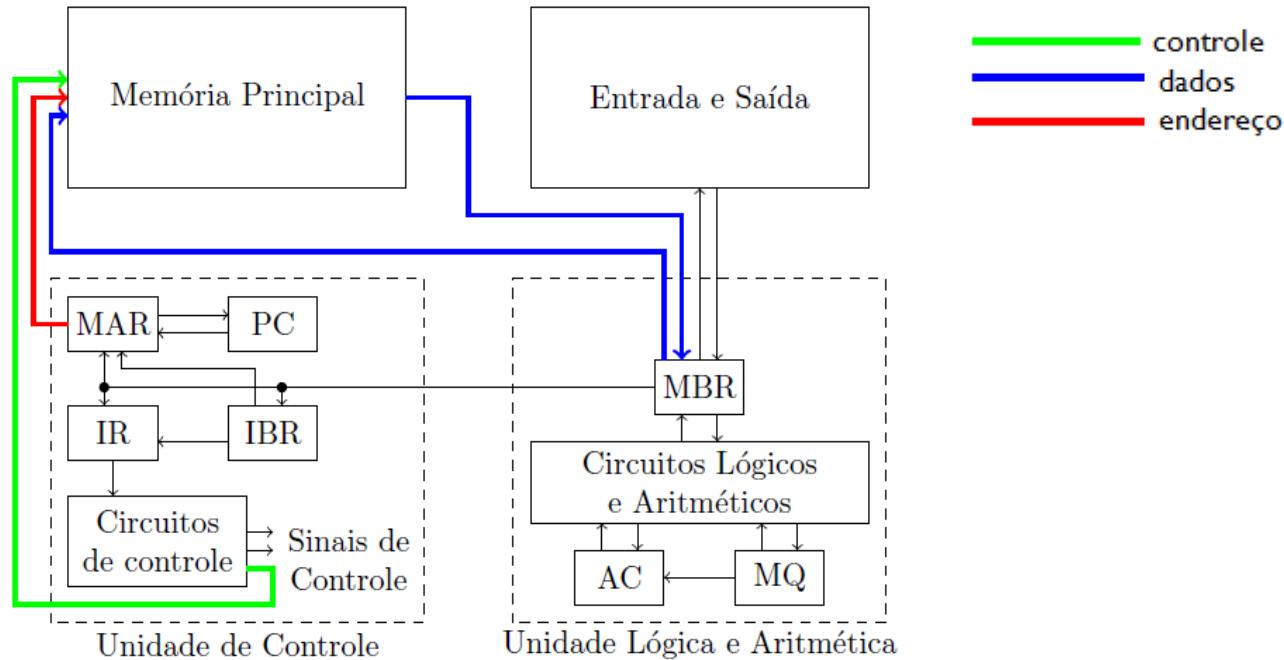
IAS – detalhes – Operação de leitura na memória



1. O endereço da palavra a ser lida é escrito no registrador MAR;
2. Os circuitos de controle da unidade de controle (UC) enviam um sinal de controle através de um canal de comunicação de controle à memória principal, solicitando a leitura do dado;
3. A memória principal lê o endereço do registrador MAR através do canal de comunicação de endereços e, de posse do endereço, lê o valor armazenado da palavra de memória associada a este endereço;
4. Por fim, a memória principal grava o valor lido no registrador MBR através do canal de comunicação de dados.

Primeira Geração - Válvulas

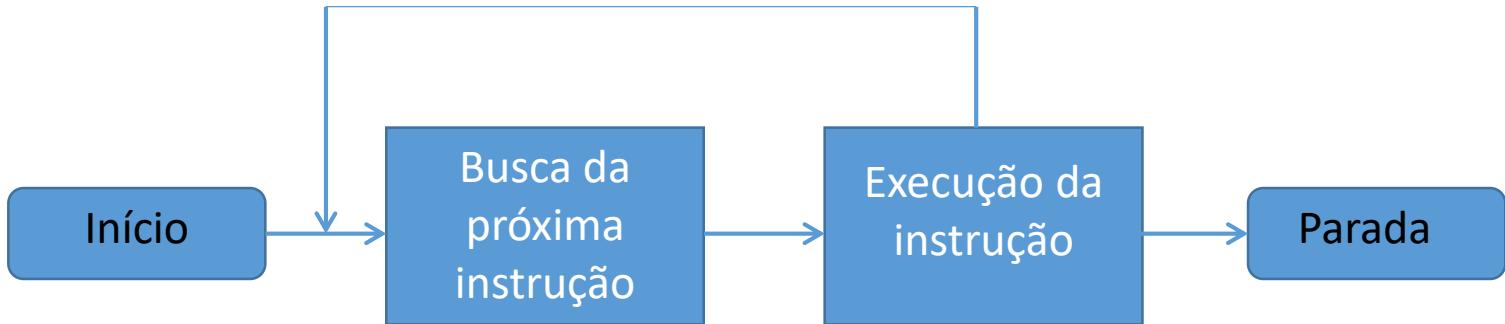
IAS – detalhes – Operação de escrita na memória



1. O endereço da palavra que armazenara o dado é escrito no registrador MAR;
2. O dado a ser armazenado é gravado no registrador MBR;
3. Os circuitos de controle da unidade de controle (UC) enviam um sinal de controle à memória principal, solicitando a escrita do dado;
4. A memória principal lê o endereço do registrador MAR através do canal de comunicação de dados, lê o dado do registrador MBR e armazena este valor na palavra de memória associada ao endereço lido de MAR;

Primeira Geração - Válvulas

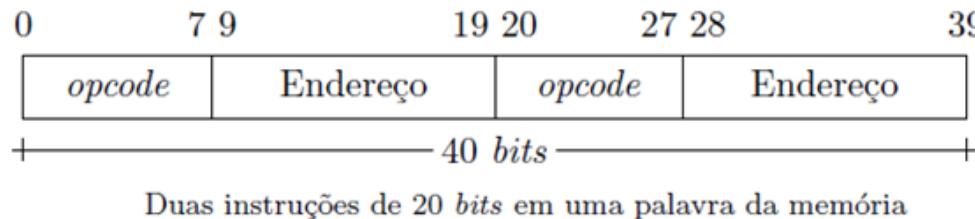
IAS – detalhes – Execução de instruções



A execução de uma instrução é realizada em dois ciclos: o “ciclo de busca” e o “ciclo de execução”. O ciclo de busca consiste em buscar a instrução da memória (ou do registrador IBR) e armazenar no IR. O ciclo de execução, por sua vez, consiste em interpretar a instrução armazenada no registrador IR e realizar as operações necessárias para execução da mesma.

Primeira Geração - Válvulas

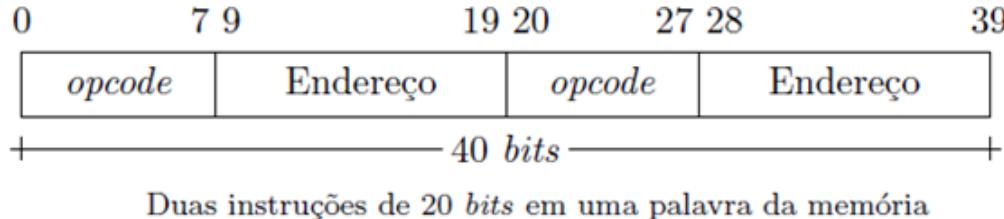
IAS – detalhes – Ciclo de busca (à esquerda)



1. A UC move o endereço em PC para MAR;
2. A UC envia um sinal de controle para a memória fazer uma operação de leitura;
3. A memória lê a palavra de memória e transfere o conteúdo para o registrador MBR;
4. A UC copia a segunda metade (bits 20 a 39) do registrador MBR e salva no registrador IBR. Estes bits correspondem à instrução à direita da palavra de memória.
5. A UC copia os 8 bits à esquerda do registrador MBR para o registrador IR. Estes bits correspondem ao campo de operação da instrução à esquerda da palavra de memória.
6. A UC copia os 12 bits subsequentes ao campo de operação (bits 8 a 19) e os transfere para o registrador MAR. Estes bits correspondem ao campo endereço da instrução e devem estar no registrador MAR caso a instrução precise acessar a memória durante o ciclo de execução.
7. A UC incrementa o valor no registrador PC, indicando que o próximo par de instruções a ser lido da memória deve ser lido do endereço PC + 1.

Primeira Geração - Válvulas

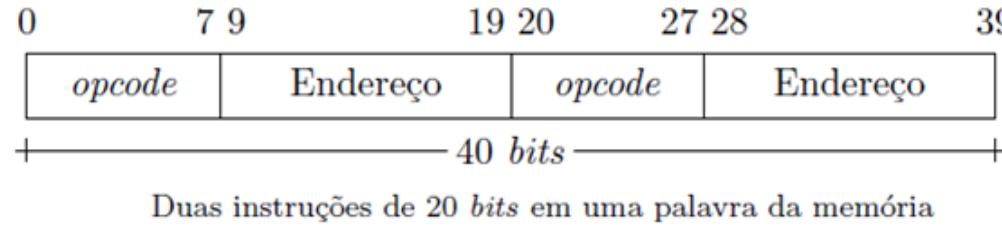
IAS – detalhes – Ciclo de busca (à direita)



1. Se a última instrução executada foi a instrução à esquerda (não houve desvio no fluxo de controle), então:
 - (a) A **UC** copia os 8 bits à esquerda do registrador **IBR** para o registrador **IR**. Estes bits correspondem ao campo de operação da instrução armazenada em **IBR**.
 - (b) A **UC** copia os 12 bits subsequentes ao campo de operação (bits 8 a 19) e os transfere para o registrador **MAR**. Estes bits correspondem ao campo endereço da instrução e devem estar no registrador **MAR** caso a instrução precise acessar a memória durante o ciclo de execução.

Primeira Geração - Válvulas

IAS – detalhes – Ciclo de busca (à direita)

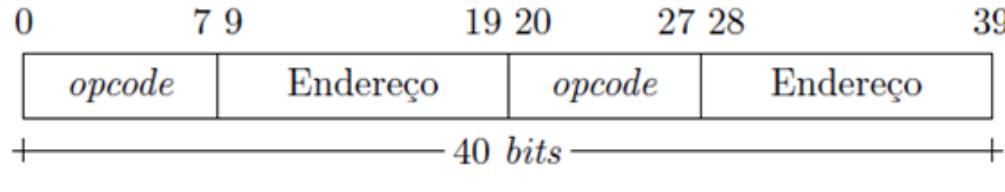


2. Senão:

- (a) A UC move o endereço em PC para MAR;
- (b) A UC envia um sinal de controle para a memória fazer uma operação de leitura;
- (c) A memória lê a palavra de memória e transfere o conteúdo para o registrador MBR;
- (d) A UC copia os bits 20 a 27 do registrador MBR para o registrador IR. Estes bits correspondem ao campo de operação da instrução à direita da palavra de memória lida.
- (e) A UC copia os 12 bits subsequentes ao campo de operação (bits 28 a 39) e os transfere para o registrador MAR. Estes bits correspondem ao campo endereço da instrução à direita da palavra de memória.
- (f) A UC incrementa o valor no registrador PC, pois a instrução à esquerda não foi executada e o mesmo não foi incrementado anteriormente.

Primeira Geração - Válvulas

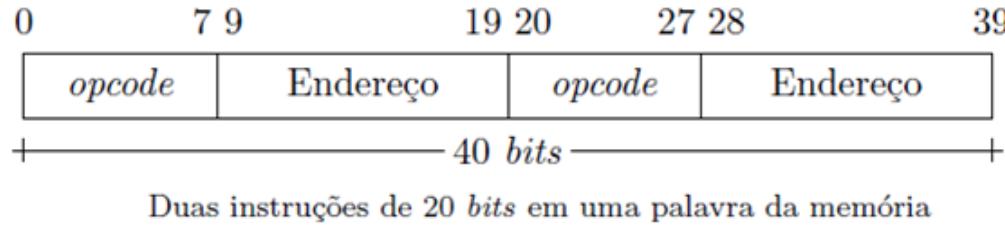
IAS – detalhes – Ciclo de execução



1. Interpretação dos bits do campo operação da instrução (*opcode*) , armazenados em IR. Esta operação é também chamada de decodificação, pois a operação a ser realizada se encontra codificada, em forma de números, dentro do campo operação.
2. Após a identificação da instrução, a UC verifica se a instrução requer a busca de operandos da memória. Se a busca for necessária, então:
 - (a) A UC envia um sinal para a memória realizar uma operação de leitura. Note que o endereço do operando já foi transferido para o registrador MAR durante o ciclo de busca.
 - (b) A memória lê a palavra de memória e transfere o conteúdo para o registrador MBR;

Primeira Geração - Válvulas

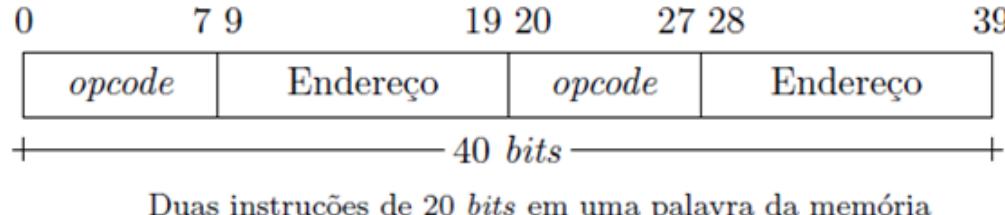
IAS – detalhes – Ciclo de execução



3. Se a instrução envolve a realização de uma operação lógica ou aritmética:
 - (a) A **UC** envia sinais de controle para a unidade lógica aritmética realizar a operação associada com a Instrução. Note que neste ponto todos os operandos da operação já se encontram em registradores na unidade lógica e aritmética.
 - (b) A **ULA** realiza a operação lógica ou aritmética de acordo com os sinais enviados pela **UC**. Estas operações incluem transferência de dados entre registradores da **ULA**, soma, subtração, multiplicação, divisão e outras.
 - (c) A **ULA** grava o resultado da operação em seus registradores: AC, MQ ou MBR.

Primeira Geração - Válvulas

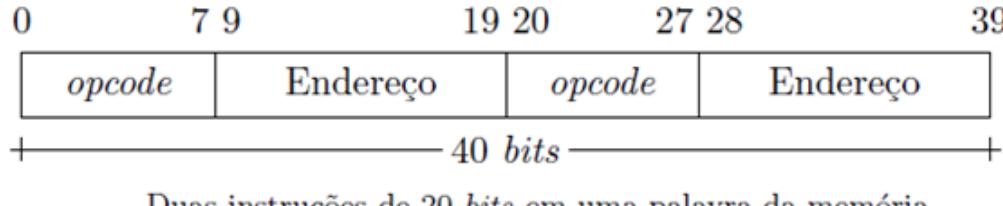
IAS – detalhes – Ciclo de execução



4. Se a instrução envolve a gravação do resultado na memória:
 - (a) A UC envia um sinal para a memória realizar uma operação de escrita. Note que o endereço do operando já foi transferido para o registrador MAR durante o ciclo de busca e o dado já foi transferido de AC para MBR no passo anterior.
 - (b) A memória lê o dado de MBR e o grava na palavra de memória associada ao endereço lido de MAR.

Primeira Geração - Válvulas

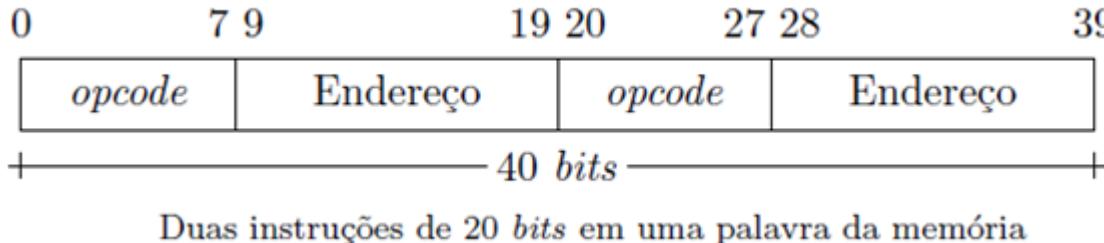
IAS – detalhes – Ciclo de execução



5. Se a execução da instrução implica no desvio do fluxo de controle, ou seja, se a instrução “salta” para uma outra instrução:
 - (a) A **UC** move o conteúdo do registrador **MAR** para **PC**. Note que o registrador **MAR** contém o valor do campo endereço da instrução sendo executada. No caso de “instruções de salto”, este campo contém o endereço da instrução para o qual o fluxo de execução deve ser desviado.
 - (b) Caso a execução corresponda a um salto para a instrução à esquerda da palavra de memória selecionada, dá-se início ao ciclo de busca de instrução à esquerda. Caso o salto seja para a instrução à direita, o ciclo de busca de instrução à direita com desvio de fluxo é iniciado.

Primeira Geração - Válvulas

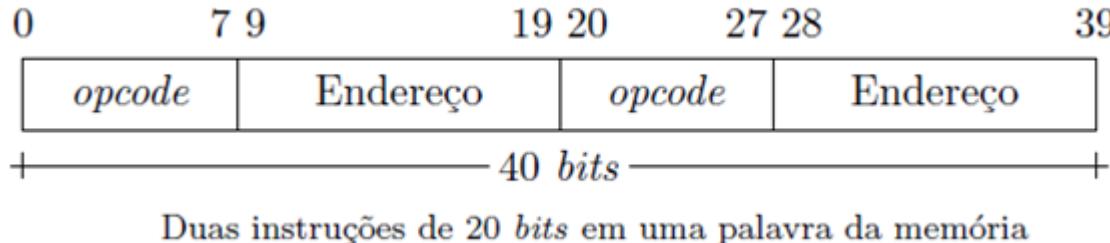
IAS – detalhes – Ciclo de execução da instrução ADD(X)



1. A UC interpreta os bits armazenados em IR (0000 0101 no caso da instrução ADD M(X)) e identifica a instrução como sendo uma soma.
 2. Após a identificação da instrução , o UC sabe que a instrução requer a busca de operandos da memória. Dessa forma:
 - (a) A UC envia um sinal para a memória realizar uma operação de leitura. Relembrando que o endereço do operando já foi transferido para o registrador MAR durante o ciclo de busca.
 - (b) A memória lê a palavra de memória e transfere o conteúdo para o registrador MBR;
 3. A UC sabe que a instrução ADD envolve a realização de uma operação de soma na ULA, então:
 - (a) A UC envia sinais para a unidade lógica e aritmética (ULA) solicitando a realização da soma dos valores armazenados em AC e MBR. Note que neste ponto todos os operandos da operação já se encontram em AC e MBR.
 - (b) A ULA realiza a operação de soma.
 - (c) A ULA grava o resultado da soma no registrador AC. ULA: AC, MQ ou MBR.
- Note que os passos 4 (armazenamento do resultado na memória) e 5 (desvio do fluxo de controle) não são necessários nesta instrução.

Primeira Geração - Válvulas

IAS – Instruções e programação

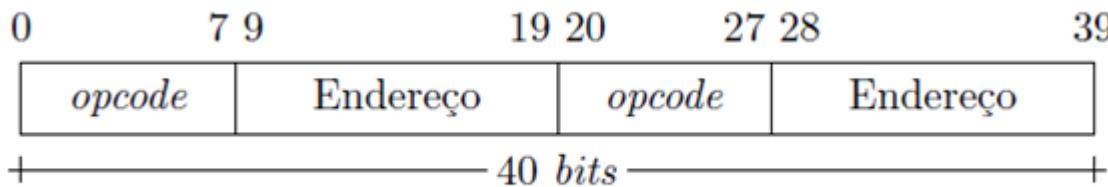


O conjunto de instruções do computador IAS possui 21 instruções. As instruções podem ser classificadas em 4 tipos distintos:

- **Transferência de dados**: instruções para mover dados entre a memória e os registradores;
- **Salto**: instruções para desviar o fluxo da execução das instruções.
- **Aritmética**: instruções para realização de operações aritméticas.
- **Modificação de endereço**: instruções para alterar o campo endereço de outras instruções.

Primeira Geração - Válvulas

IAS – Instruções de transferência de dados



Duas instruções de 20 bits em uma palavra da memória

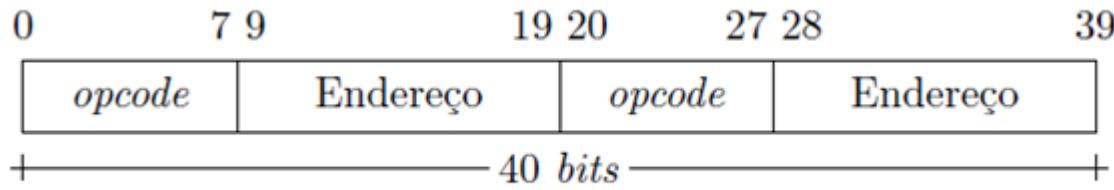
➤ Instrução LOAD M(X)

Sintaxe	Operação	Codificação
LOAD M(X)	AC := Mem[X]	00000001 X

Transfere o valor armazenado no endereço X da memória para o registrador AC.

Primeira Geração - Válvulas

IAS – Instruções de transferência de dados



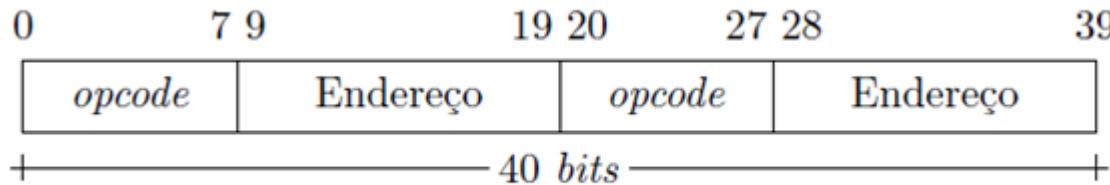
Duas instruções de 20 bits em uma palavra da memória

➤ Instrução LOAD MQ,M(X)

Sintaxe	Operação	Codificação
LOAD MQ,M(X)	MQ := Mem[X]	00001001 X
Transfere o valor armazenado no endereço X da memória para o registrador MQ.		

Primeira Geração - Válvulas

IAS – Instruções de transferência de dados



Duas instruções de 20 bits em uma palavra da memória

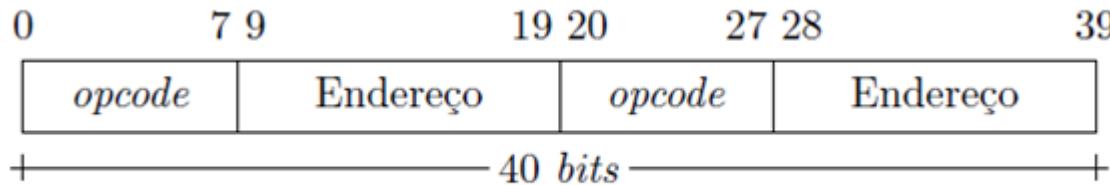
➤ Instrução STOR M(X)

Sintaxe	Operação	Codificação
STOR M(X)	Mem[X] := AC	00100001 X

Transfere o conteúdo do registrador AC para a palavra da memória no endereço X.

Primeira Geração - Válvulas

IAS – Instruções de transferência de dados



Duas instruções de 20 bits em uma palavra da memória

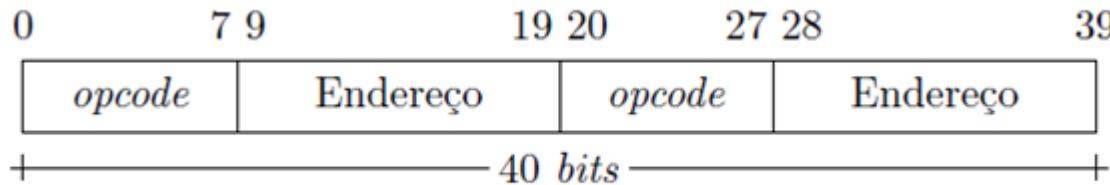
➤ Instrução LOAD MQ

Sintaxe	Operação	Codificação
LOAD MQ	AC := MQ	00001010 -

Transfere o conteúdo do registrador MQ para o registrador AC.

Primeira Geração - Válvulas

IAS – Instruções de transferência de dados



Duas instruções de 20 bits em uma palavra da memória

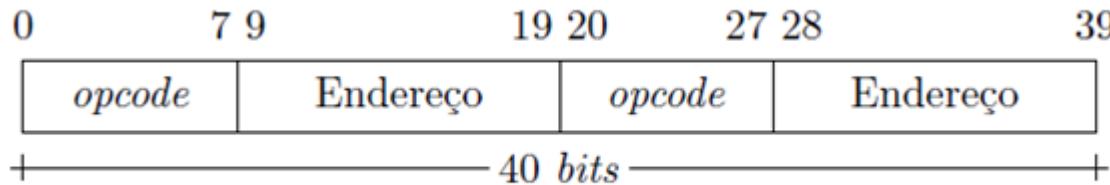
➤ Instrução LOAD -M(X)

Sintaxe	Operação	Codificação
LOAD -M(X)	AC := -(Mem[X])	00000010 X

Transfere o negativo do valor armazenado no endereço X da memória para o registrador AC.

Primeira Geração - Válvulas

IAS – Instruções aritméticas



Duas instruções de 20 bits em uma palavra da memória

➤ Instrução ADD M(X)

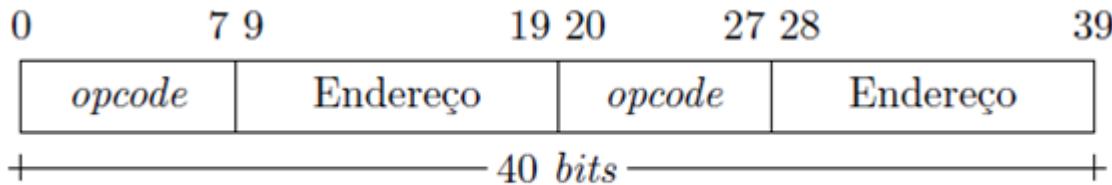
Sintaxe	Operação	Codificação
ADD M(X)	AC := AC + Mem[X]	00000101 X

Soma o valor armazenado no endereço X da memória com o valor armazenado no registrador AC e armazena o resultado no registrador AC.

```
LOAD M(100)    # AC := Mem[100]
ADD M(101)     # AC := AC + Mem[101]
STOR M(102)    # Mem[102] := AC
```

Primeira Geração - Válvulas

IAS – Instruções aritméticas



Duas instruções de 20 bits em uma palavra da memória

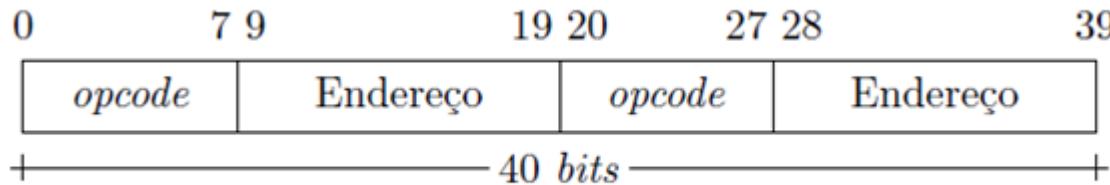
➤ Instrução ADD |M(X)|

Sintaxe	Operação	Codificação		
ADD M(X)	$AC := AC + Mem[X] $	<table border="1"><tr><td>00000111</td><td>X</td></tr></table>	00000111	X
00000111	X			

Soma o absoluto do valor armazenado no endereço X da memória com o valor armazenado no registrador AC e armazena o resultado no registrador AC.

Primeira Geração - Válvulas

IAS – Instruções aritméticas



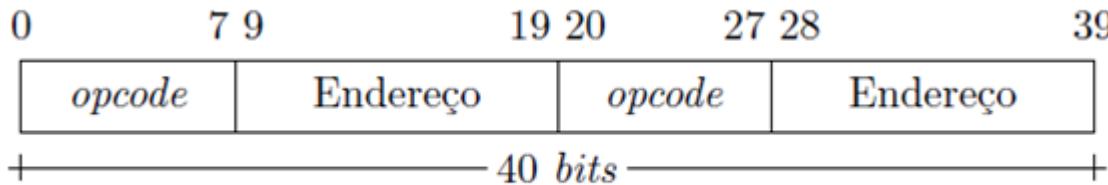
Duas instruções de 20 bits em uma palavra da memória

➤ Instrução SUB M(X)

Sintaxe	Operação	Codificação
SUB M(X)	$AC := AC - \text{Mem}[X]$	00000110 X
Subtrai o valor armazenado no endereço X da memória do valor armazenado no registrador AC e armazena o resultado no registrador AC.		

Primeira Geração - Válvulas

IAS – Instruções aritméticas



Duas instruções de 20 bits em uma palavra da memória

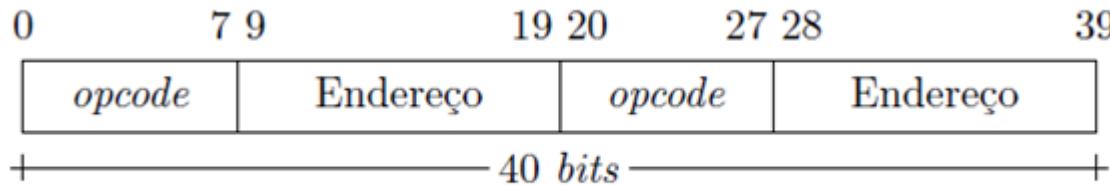
➤ Instrução SUB |M(X)|

Sintaxe	Operação	Codificação
SUB M(X)	AC := AC - Mem[X]	00001000 X

Subtrai o absoluto do valor armazenado no endereço X da memória do valor armazenado no registrador AC e armazena o resultado no registrador AC.

Primeira Geração - Válvulas

IAS – Instruções aritméticas



Duas instruções de 20 bits em uma palavra da memória

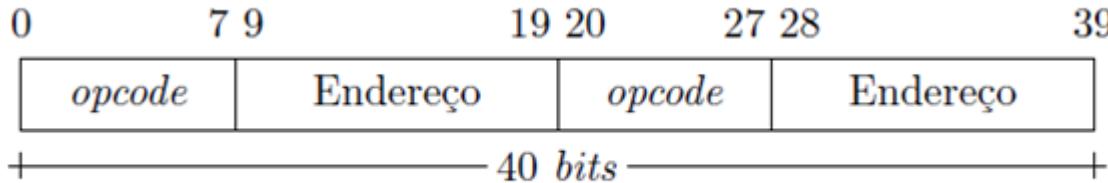
➤ Instrução MUL M(X)

Sintaxe	Operação	Codificação
MUL M(X)	AC:MQ := MQ * Mem[X]	00001011 X

Multiplica o valor no endereço X da memória pelo valor em MQ e armazena o resultado em AC e MQ. O resultado é um número de 80 bits. Os 40 bits mais significativos são armazenados em AC e os 40 bits menos significativos são armazenados em MQ.

Primeira Geração - Válvulas

IAS – Instruções aritméticas



Duas instruções de 20 bits em uma palavra da memória

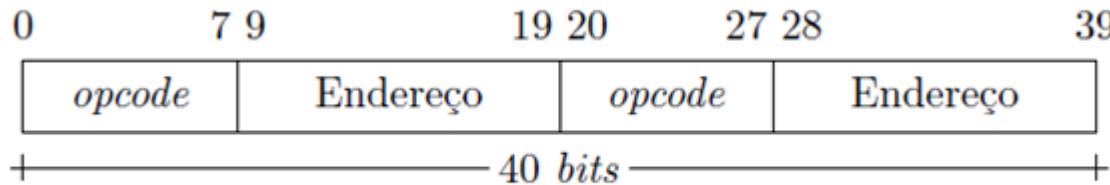
➤ Instrução DIV M(X)

Sintaxe	Operação	Codificação
DIV M(X)	$MQ := AC / Mem[X]$ $AC := AC \% Mem[X]$	00001100 X

Divide o valor em AC pelo valor armazenado no endereço X da memória.
Coloca o quociente em MQ e o resto em AC.

Primeira Geração - Válvulas

IAS – Instruções aritméticas



Duas instruções de 20 bits em uma palavra da memória

➤ Instrução RSH

Sintaxe	Operação	Codificação
RSH	$AC := AC \gg 1$	00010101 X

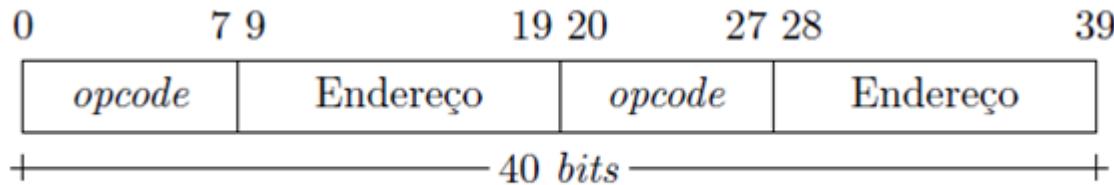
Desloca os bits do registrador AC para a direita.

$$0000\ 0101_2(5_{10}) \gg 1 = 0000\ 0010_2(2_{10})$$

$$0000\ 1000_2(8_{10}) \gg 1 = 0000\ 0100_2(4_{10})$$

Primeira Geração - Válvulas

IAS – Instruções aritméticas



Duas instruções de 20 bits em uma palavra da memória

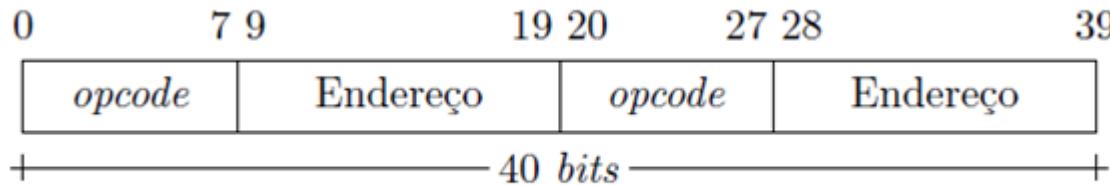
➤ Instrução LSH

Sintaxe	Operação	Codificação
LSH	$AC := AC \ll 1$	00010100 X

Desloca os bits do registrador AC para a esquerda.

Primeira Geração - Válvulas

IAS – Instruções de salto



Duas instruções de 20 bits em uma palavra da memória

➤ Instrução JUMP M(X,0:19)

Sintaxe	Operação	Codificação
JUMP M(X,0:19)	PC := M(X) e executa instrução à esquerda.	00001101 X

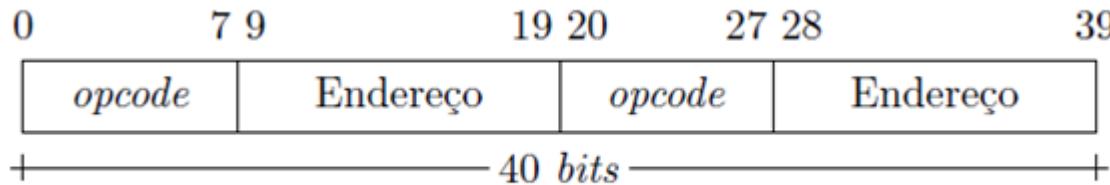
Salta para a instrução à esquerda (*bits* 0 a 19) da palavra de memória armazenada no endereço M(X).

000 LOAD M(100);
001 STOR M(100);

ADD M(101)
JUMP M(000,0:19)

Primeira Geração - Válvulas

IAS – Instruções de salto



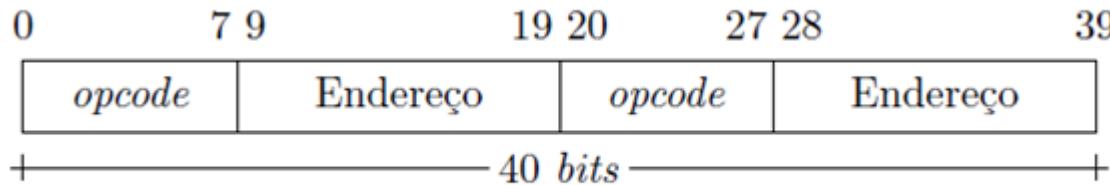
Duas instruções de 20 bits em uma palavra da memória

➤ Instrução JUMP M(X,20:39)

Sintaxe	Operação	Codificação
JUMP M(X,20:39)	PC := M(X) e executa instrução à direita.	00001110 X
Salta para a instrução à direita (<i>bits 20 a 39</i>) da palavra de memória armazenada no endereço M(X).		

Primeira Geração - Válvulas

IAS – Instruções de salto



Duas instruções de 20 bits em uma palavra da memória

➤ Instrução JUMP+ M(X,0:19)

Sintaxe	Operação	Codificação
JUMP+ M(X,0:19)	Se $AC \geq 0$ então $PC := M(X)$ e executa instrução à esquerda, senão, executa a próxima instrução.	00001111 X
Salta para a instrução à esquerda (<i>bits</i> 0 a 19) da palavra de memória se o valor armazenado em AC for maior ou igual à zero.		

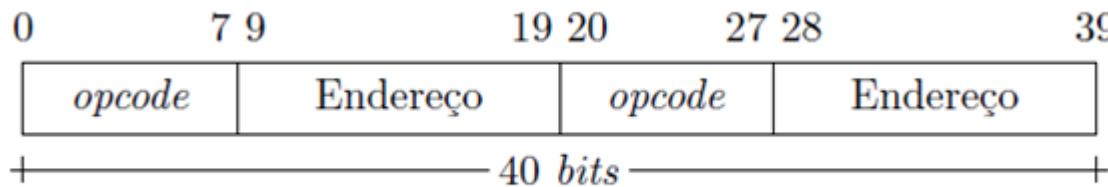
```

000 LOAD M(100);      ADD M(0101)
001 STOR M(100);      LOAD M(0102)
002 SUB  M(103);      STOR M(0102)
003 JUMP+ M(000,0:19); ...

102 00 00 00 00 09 # Contador
103 00 00 00 00 01 # Constante 1
  
```

Primeira Geração - Válvulas

IAS – Instruções de salto



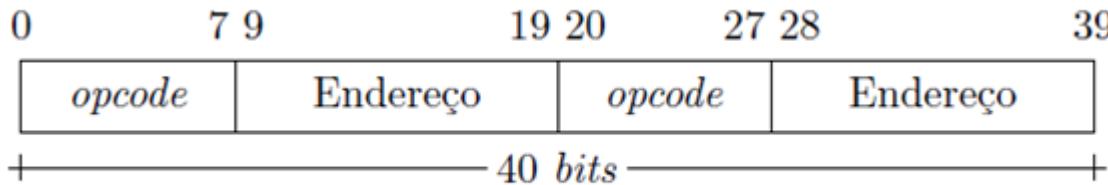
Duas instruções de 20 bits em uma palavra da memória

➤ Instrução JUMP+ M(X,20:39)

Sintaxe	Operação	Codificação
JUMP+ M(X,20:39)	Se $AC \geq 0$ então $PC := M(X)$ e executa instrução à direita, senão, executa a próxima instrução.	00010000 X
Salta para a instrução à direita (<i>bits 20 a 39</i>) da palavra de memória se o valor armazenado em AC for maior ou igual à zero.		

Primeira Geração - Válvulas

IAS – Instruções de salto



Duas instruções de 20 bits em uma palavra da memória

➤ Instrução JUMP+ M(X,20:39)

Sintaxe	Operação	Codificação
JUMP+ M(X,20:39)	Se $AC \geq 0$ então $PC := M(X)$ e executa instrução à direita, senão, executa a próxima instrução.	00010000 X

Salta para a instrução à direita (bits 20 a 39) da palavra de memória se o valor armazenado em AC for maior ou igual à zero.

```

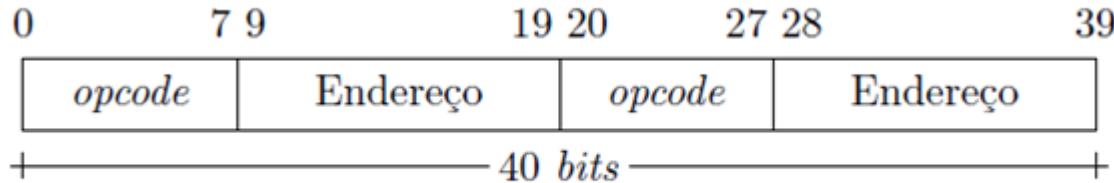
000 LOAD M(100);           SUB M(101)      # Se Y > X
001 JUMP+ M(003,0:19);    LOAD M(101)      # Z = Y
002 STOR M(102);          JUMP M(004,0:19) # senão
003 LOAD M(100);          STOR M(102)     # Z = X
004 ...

```

100 00 00 00 00 01	# Variável X
101 00 00 00 00 02	# Variável Y
102 00 00 00 00 00	# Variável Z

Primeira Geração - Válvulas

IAS – Instruções de modificação de endereço

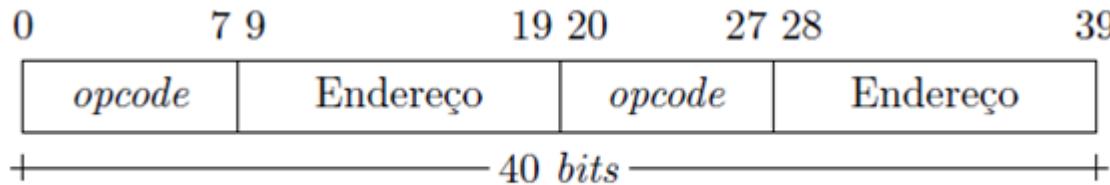


Duas instruções de 20 *bits* em uma palavra da memória

```
int A[1024];  
  
int soma_elementos()  
{  
    int i;  
    int soma=0;  
    for (i=0; i<1024; i++)  
        soma = soma + A[i];  
    return soma;  
}
```

Primeira Geração - Válvulas

IAS – Instruções de modificação de endereço



Duas instruções de 20 bits em uma palavra da memória

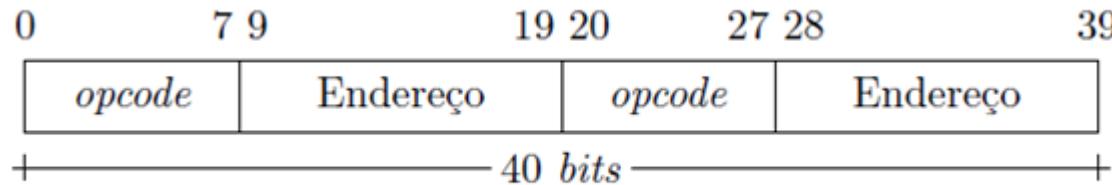
➤ Instrução STOR M(X,8:19)

Sintaxe	Operação	Codificação
STOR M(X,8:19)	Mem[X](8:19) := AC(28:39)	00010010 X

Move os 12 bits à direita do registrador AC para o campo endereço da instrução à esquerda da palavra de memória no endereço X.

Primeira Geração - Válvulas

IAS – Instruções de modificação de endereço



Duas instruções de 20 *bits* em uma palavra da memória

➤ Instrução STOR M(X,28:39)

Sintaxe	Operação	Codificação
STOR M(X,28:39)	Mem[X](28:39) := AC(28:39)	00010011 X

Move os 12 *bits* à direita do registrador AC para o campo endereço da instrução à direita da palavra de memória no endereço X.

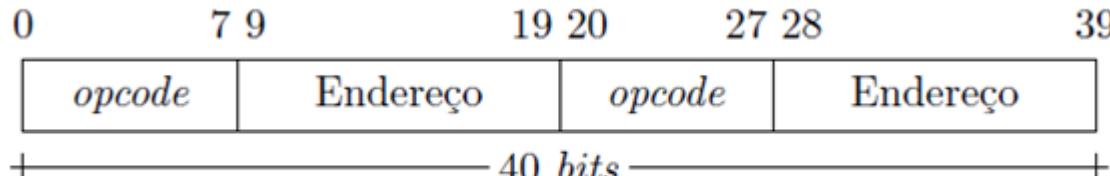
Primeira Geração - Válvulas

IAS – Exercício

Elaborar um programa, em código do IAS, que realize a soma dos elementos de um vetor de 20 números armazenados a partir do endereço 100 h.

Primeira Geração - Válvulas

IAS – Resolução do exercício



Duas instruções de 20 bits em uma palavra da memória

	Endereço	Instruções / Dados	
1			
2			
3	000	LOAD M(OF2); STOR M(002,28:39) # Modifica o endereço da instrução ADD	
4	001	ADD M(OF1); STOR M(OF2)	# e atualiza o apontador.
5	002	LOAD M(OF3); ADD M(000)	# Carrega a variável e soma com o conteúdo
6			# do vetor apontado pelo apontador.
7	003	STOR M(OF3); LOAD M(OF0)	# Salva a soma e carrega o contador de it.
8	004	SUB M(OF1); STOR M(OF0)	# Atualiza o contador de iterações.
9	005	JUMP+ M(000,0:19); ...	
10			
11	OF0	00 00 00 00 00 19	# Contador de iterações
12	OF1	00 00 00 00 00 01	# Constante 1
13	OF2	00 00 00 01 00	# Apontador
14	OF3	00 00 00 00 00	# variável soma
15			
16	100	00 00 00 00 00 00	# Primeiro elemento do vetor
17	

Primeira Geração - Válvulas

IAS – Exercícios proposto nº 1

Elaborar um programa, em código do IAS, que dado um vetor de 20 números armazenados a partir do endereço 100, determine o maior e o menor elemento do vetor e armazene o resultado no endereço 200 e 201, respectivamente.

Primeira Geração - Válvulas

IAS – Linguagem de montagem do IAS

Montador: é uma ferramenta que converte código em linguagem de montagem para código em linguagem de máquina. :

LOAD M(0x102)	 Montador	01	10	20	B1	03
MUL M(0x103)		0A	00	02	11	02
LOAD MQ						
STOR M(0x102)						

Primeira Geração - Válvulas

IAS – Linguagem de montagem do IAS

Montador: é uma ferramenta que converte código em linguagem de montagem para código em linguagem de máquina. :

LOAD M(0x102)	Montador	01 10 20 B1 03
MUL M(0x103)		0A 00 02 11 02
LOAD MQ		
STOR M(0x102)		

Primeira Geração - Válvulas

IAS – Linguagem de montagem do IAS

➤ A diretiva .org

A diretiva **.org** informa ao montador o endereço de memória onde o montador deve iniciar (ou continuar) a geração do código.

1	.org 0x000	000	01	10	20	B1	03
2	LOAD M(0x102)	001	0A	00	00	D0	20
3	MUL M(0x103)						
4	LOAD MQ	020	21	10	20	00	00
5	JUMP M(0x020,0:19)						
6							
7	.org 0x020						
8	STOR M(0x102)						
9							
10	Ling. de Montagem		Mapa de memória				

Primeira Geração - Válvulas

IAS – Linguagem de montagem do IAS

➤ A diretiva .word

A diretiva **.word** é uma diretiva que auxilia o programador a adicionar dados à memória. Para adicionar um dado, basta inserir a diretiva **.word** e um valor de 40 bits no programa.

```
1      .org 0x102          000  01 10 20 B1 03
2          .word 0x1        001  0A 00 00 D0 00
3          .word 10         102  00 00 00 00 00 01
4
5      .org 0x000          103  00 00 00 00 00 OA
6          LOAD M(0x102)
7          MUL  M(0x103)
8          LOAD MQ
9          JUMP M(0x000,0:19)
10
11
```

Primeira Geração - Válvulas

IAS – Linguagem de montagem do IAS

➤ Rótulos

Rótulos são anotações no código que serão convertidas em endereços pelo montador. A sintaxe de um rótulo é uma palavra terminada com o caractere ":" (dois pontos). Podem ser utilizados para especificar um local no código para onde uma instrução de desvio deve saltar. O código abaixo utiliza um rótulo (laco:) para representar o alvo de uma instrução de salto.

Exemplos:

```
1 laco:  
2   LOAD M(0x100)  
3   SUB  M(0x200)  
4   JUMP M(laco)
```

Primeira Geração - Válvulas

IAS – Linguagem de montagem do IAS

➤ Rótulos

Rótulos são anotações no código que **serão convertidas em endereços pelo montador**. A sintaxe de um rótulo é uma palavra terminada com o caractere ":" (dois pontos). Podem ser utilizados para especificar um local no código para onde uma instrução de desvio deve saltar. O código abaixo utiliza um rótulo (laco:) para representar o alvo de uma instrução de salto.

Exemplos:

```
1 .org 0x000
2 laco:
3   LOAD M(var_x)
4   SUB  M(var_y)
5   JUMP M(laco)
6 .org 0x100
7 var_x:
8 .org 0x200
9 var_y:
```

Primeira Geração - Válvulas

IAS – Linguagem de montagem do IAS

➤ Rótulos

Podem ser utilizados em conjunto com a diretiva `.word` para declarar variáveis ou constantes. Em vez de associar um rótulo a um endereço fixo de memória, pode-se declarar um rótulo e logo em seguida adicionar um dado neste endereço de memória com a diretiva `.word`, e o próximo rótulo, se usado, conterá o endereço da próxima palavra da memória. O trecho de código a seguir mostra exemplos de declaração de variáveis e constantes. Neste caso, o rótulo **var x** será associado ao endereço de memória 0x100 e o rótulo **const1** será associado ao endereço de memória 0x101.

Exemplos:

```
1      .org 0x000          000  01 10 00 61 01
2      laco:              001  0D 00 00 00 00
3          LOAD M(var_x)
4          SUB  M(const1)    100  00 00 00 00 09
5          JUMP M(laco)     101  00 00 00 00 01
6
7      .org 0x100
8      var_x:
9          .word 00 00 00 00 09
10     const1:
11        .word 00 00 00 00 01
12
13     Linguagem de Montagem      Mapa de memória
```

Primeira Geração - Válvulas

IAS – Linguagem de montagem do IAS

Rótulos

O trecho de código a seguir mostra um exemplo onde o rótulo vetor é utilizado em conjunto com a diretiva `.word` para adicionar o endereço base do vetor à palavra da memória associada com o endereço do rótulo base. Em outras palavras, declaramos a variável base e a inicializamos com o valor 0x100, ou seja, o endereço inicial do vetor.

Exemplos:

```
1      .org 0x100          100  00 00 00 01 01  
2      base:             101  00 00 00 00 00  
3          .word vetor   102  00 00 00 00 01  
4      vetor:            103  00 00 00 00 02  
5          .word 00 00 00 00 00  
6          .word 00 00 00 00 01  
7          .word 00 00 00 00 02  
8      fim_vetor:  
9  
10
```

Primeira Geração - Válvulas

IAS – Linguagem de montagem do IAS

➤ Diretiva .align N

Esta diretiva informa ao montador para continuar a montagem a partir da próxima palavra com endereço múltiplo de N.

Exemplos:

```
1 .org 0x000
2 laco:
3 LOAD M(var_x)
4 SUB M(var_y)
5 JUMP M(laco)
6 var_x: .word 0x1
7 var_y: .word 0x2
```

1	000	01	00	20	60	03
2	001	0D	00	00	00	00
3	002	00	00	00	00	01
4	003	00	00	00	00	02

```
1 .org 0x000
2 laco:
3 LOAD M(var_x)
4 SUB M(var_y)
5 JUMP M(laco)
6 .align 1
7 var_x: .word 0x1
8 var_y: .word 0x2
```

Primeira Geração - Válvulas

IAS – Linguagem de montagem do IAS

➤ Diretiva .wfill

Esta diretiva preenche N palavras da memória com o dado D.

Exemplos:

```
1 .org 0x000
2 laco:
3   JUMP M(laco)
4 .align 1
5 vetor:
6   .word 0x4
7   .word 0x0
8   .word 0x4
```

```
1 .org 0x000
2 laco:
3   JUMP M(laco)
4 .align 1
5 vetor:
6   .wfill 1000, 0x5
```

Primeira Geração - Válvulas

IAS – Linguagem de montagem do IAS

➤ Diretiva .set NOME VALOR

A diretiva .set NOME VALOR é a diretiva utilizada na linguagem de montagem do IAS para associar valores a nomes.

```
1      .set CODIGO 0x000          000 0D 00 00 00 00  
2      .set DADOS 0x100          100 00 00 00 00 05  
3      .set TAMANHO 200          101 00 00 00 00 05  
4                      ...  
5      .org CODIGO             1C6 00 00 00 00 05  
6      laco:  
7          JUMP M(laco)  
8  
9      .org DADOS  
10     vetor:  
11         .wfill TAMANHO, 0x5  
12  
13     Ling. de Montagem           Mapa de memória
```

```
1      .org 0x000          000 0D 00 00 00 00  
2      laco:  
3          JUMP M(laco)          100 00 00 00 00 05  
4                      ...  
5      .org 0x100             101 00 00 00 00 05  
6      vetor:  
7         .wfill 200, 0x5          1C6 00 00 00 00 05  
8  
9      Ling. de Montagem           Mapa de memória
```

Primeira Geração - Válvulas

IAS – Linguagem de montagem do IAS

➤ Exercício proposto nº 2

Elaborar um programa, em código do IAS, que dado um vetor de 10 números armazenados a partir do endereço 100, determine o número de elementos primos do vetor e armazene o resultado no endereço 300. Converter o código em linguagem de montagem para código em linguagem de máquina.

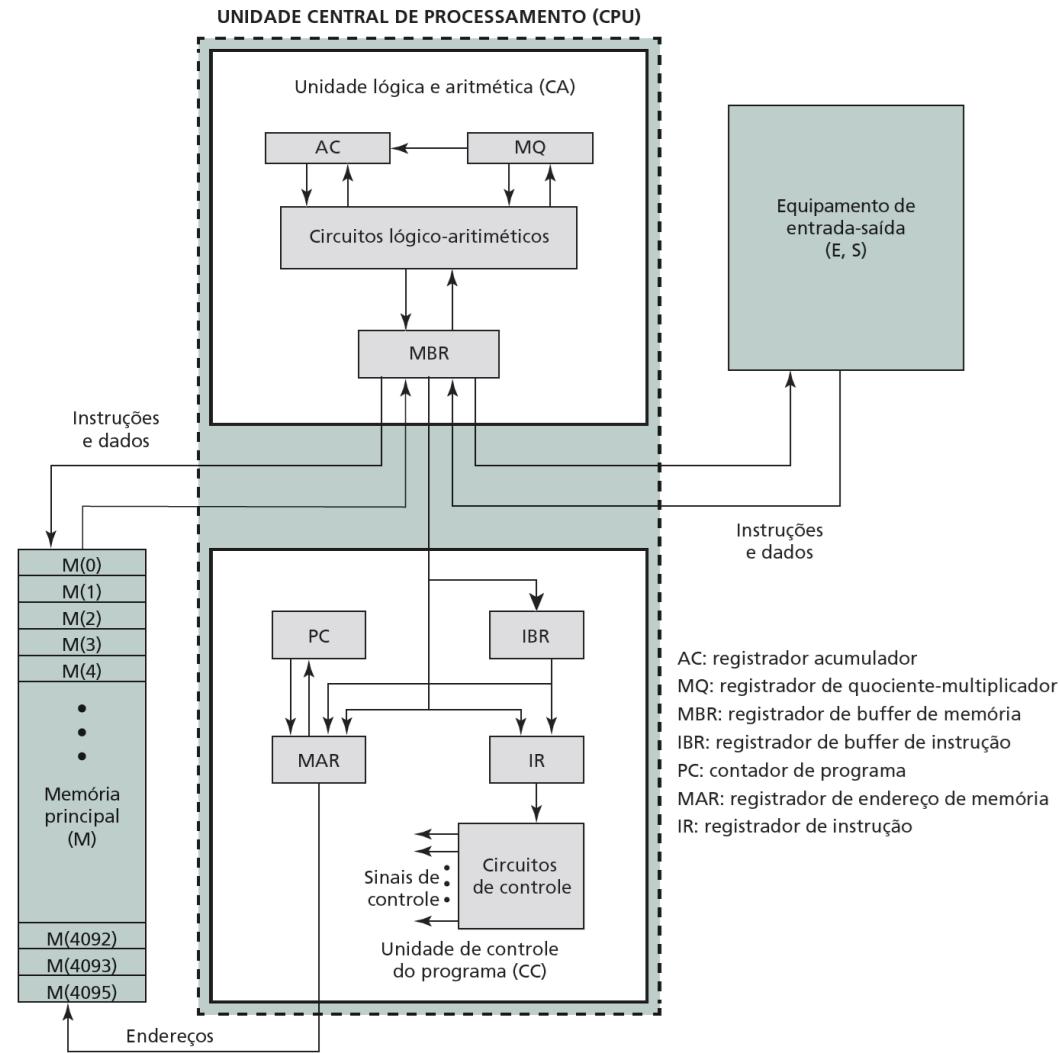
Uma breve história dos computadores

A primeira geração: válvulas

- O mais famoso computador de primeira geração é conhecido como computador **IAS**.
- A figura a seguir mostra sua estrutura. Ela consiste em:
- **Uma memória principal.**
- **Uma unidade lógica e aritmética (ALU).**
- **Uma unidade de controle.**
- **Equipamento de entrada/saída (E/S).**

Uma breve história dos computadores

A primeira geração: válvulas



Uma breve história dos computadores

A segunda geração: transistores

- A primeira mudança principal no computador eletrônico vem com a substituição das válvulas pelo transistor.
- O transistor, que é menor, mais barato e gera menos calor do que a válvula, pode ser usado da mesma maneira que uma válvula para construir computadores.
- Ao contrário da válvula, que requer fios, placas de metal e cápsula de vidro, além de vácuo, o transistor é um dispositivo de estado sólido, feito de silício.

Uma breve história dos computadores

A segunda geração: transistores

- A segunda geração viu uma introdução de unidades aritméticas e lógicas e unidades de controle, o uso de linguagem de programação de alto nível e a disponibilização dos softwares de sistema com o computador.
- Tornou-se amplamente aceito [classificar os computadores em gerações com base na tecnologia nos fundamentos de hardware empregados](#).
- Veja figura a seguir.

Uma breve história dos computadores

A segunda geração: transistores

- Gerações de computador:

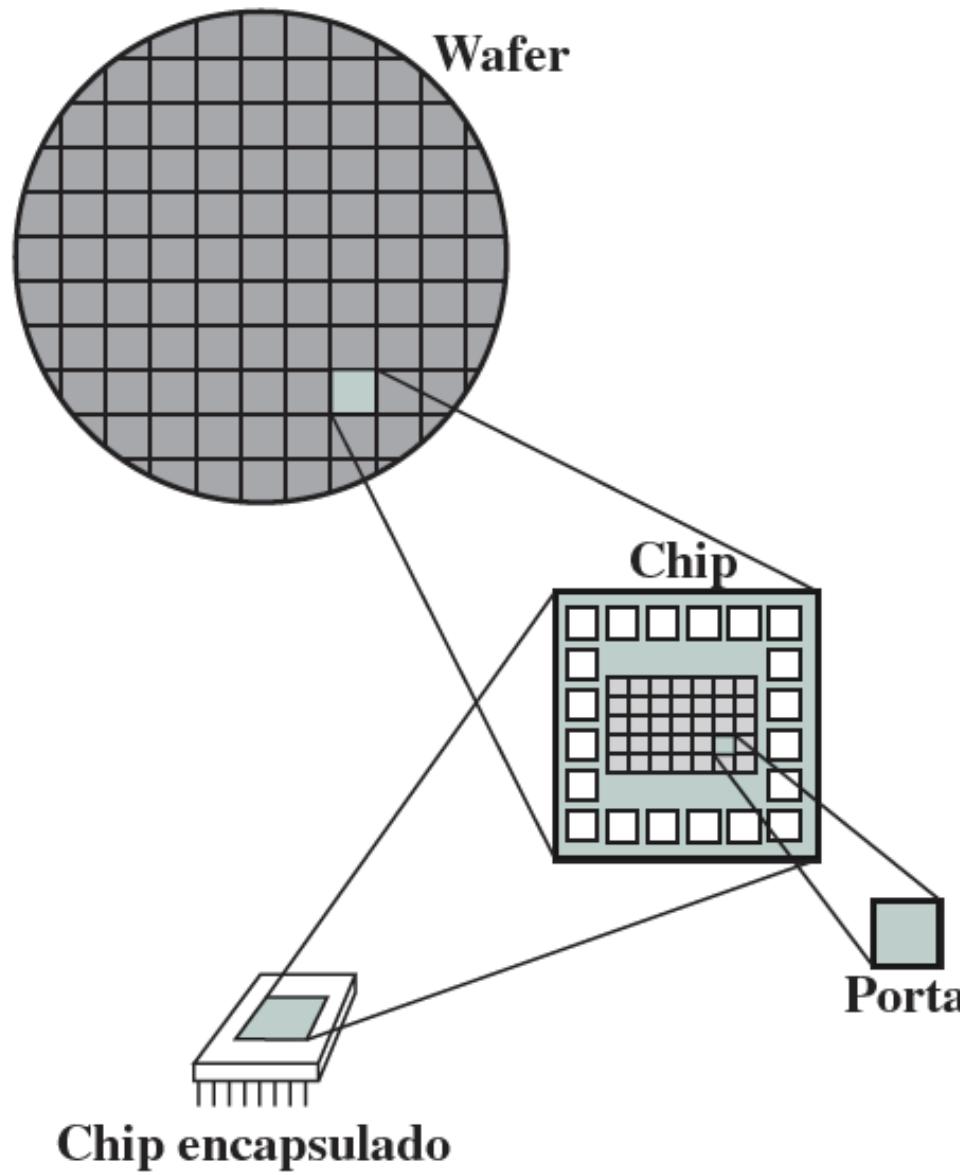
Geração	Datas aproximadas	Tecnologia	Velocidade normal (operações por segundo)
1	1946–1957	Válvula	40.000
2	1957–1964	Transistor	200.000
3	1965–1971	Integração em pequena e média escala	1.000.000
4	1972–1977	Integração em grande escala	10.000.000
5	1978–1991	Integração em escala muito grande	100.000.000
6	1991–	Integração de escala ultra grande	> 1.000.000.000

Uma breve história dos computadores

A terceira geração: circuitos integrados

- Em 1958, chegou a realização que revolucionou a eletrônica e iniciou a era da microeletrônica: **a invenção do circuito integrado**.
- A figura a seguir representa os conceitos-chave de um circuito integrado.
- Os primeiros circuitos integrados são conhecidos como **integração em pequena escala (SSI** — do inglês, *Small-Scale Integration*).

Uma breve história dos computadores



Uma breve história dos computadores

A terceira geração: circuitos integrados

- Por volta de 1964, a IBM anunciou o [System/360](#), uma nova família de produtos de computador.
- O conceito de [uma família de computadores compatíveis](#) foi moderno e extremamente bem-sucedido.
- As características de uma família são as seguintes:
 - Conjunto de instruções semelhante ou idêntico
 - Sistema operacional semelhante ou idêntico

Uma breve história dos computadores

A terceira geração: circuitos integrados

- Maior velocidade.
- Número cada vez maior de portas de E/S.
- Aumento do tamanho de memória.
- Maior custo.

Como esse conceito de família poderia ser implementado?

Uma breve história dos computadores

A terceira geração: circuitos integrados

- As diferenças foram conseguidas com base em três fatores:
 1. velocidade básica,
 2. tamanho e
 3. grau de simultaneidade
- O [System/360](#) não apenas ditou o curso futuro da IBM, mas também teve um impacto profundo sobre a indústria inteira.
- Muitos de seus recursos tornaram-se padrão de outros computadores grandes.

Uma breve história dos computadores

Gerações posteriores

- Com a introdução da **integração em grande escala (LSI)**, mais de 1.000 componentes podem ser colocados em um único chip de circuito integrado.
- A **integração em escala muito grande (VLSI** — do inglês, *Very-Large-Scale Integration*) alcançou mais de 10.000 componentes por chip, enquanto os chips com **integração em escala ultragrande (ULSI** — do inglês, *Ultra-Large-Scale Integration*) podem conter mais de um bilhão de componentes.

A evolução da arquitetura Intel x86

- As propostas dos x86 atuais representam os resultados de décadas de esforço de projeto em computadores com conjunto complexo de instruções (**CISC** — do inglês, *Complex Instruction Set Computers*).
- Uma técnica alternativa para o projeto do processador é o computador com conjunto de instruções reduzido (**RISC** — do inglês, *Reduced Instruction Set Computers*).
- A arquitetura ARM é usada em uma grande variedade de sistemas embarcados e é um dos sistemas baseados em RISC mais poderosos e bem projetados no mercado.

A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **8080:** o primeiro microprocessador de propósito geral do mundo. Esta era uma máquina de 8 bits, com um caminho de dados de 8 bits para a memória. O 8080 foi usado no primeiro computador pessoal, o Altair.
- **8086:** uma máquina muito mais poderosa, de 16 bits. Além de um caminho de dados mais largo e registradores maiores, o 8086 ostentava uma cache de instruções, ou fila, que fazia a pré-busca de algumas instruções antes que fossem executadas. Uma variante desse processador, o 8088 foi usado no primeiro computador pessoal da IBM, assegurando o sucesso da Intel. O 8086 é o primeiro aparecimento da arquitetura x86.

A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **80286:** esta extensão do 8086 permitia o endereçamento de uma memória de **16 MB**, em vez de apenas **1 MB**.
- **80386:** a primeira máquina de 32 bits da Intel e uma maior reformulação geral do produto. Com uma arquitetura de 32 bits, o 80386 competia em complexidade e potência com os minicomputadores e mainframes introduzidos alguns anos antes. **Este foi o primeiro processador da Intel a aceitar multitarefa, significando que poderia executar vários programas ao mesmo tempo.**

A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **80486:** o 80486 introduziu **o uso de tecnologia de cache muito mais sofisticada e poderosa**, bem como **pipeline sofisticado de instrução**. O 80486 também **ofereceu um coprocessador matemático embarcado**, tirando da CPU principal operações matemáticas complexas.
- **Pentium:** com o Pentium, a Intel introduziu o uso de **técnicas superescalares**, que permitem que múltiplas instruções sejam executadas em paralelo.
- **Pentium Pro:** o Pentium Pro continuou o movimento em direção à organização superescalar, iniciada com o Pentium, com o uso agressivo de **renomeação de registrador, previsão de desvio, análise de fluxo de dados e execução especulativa**.

A evolução da arquitetura Intel x86

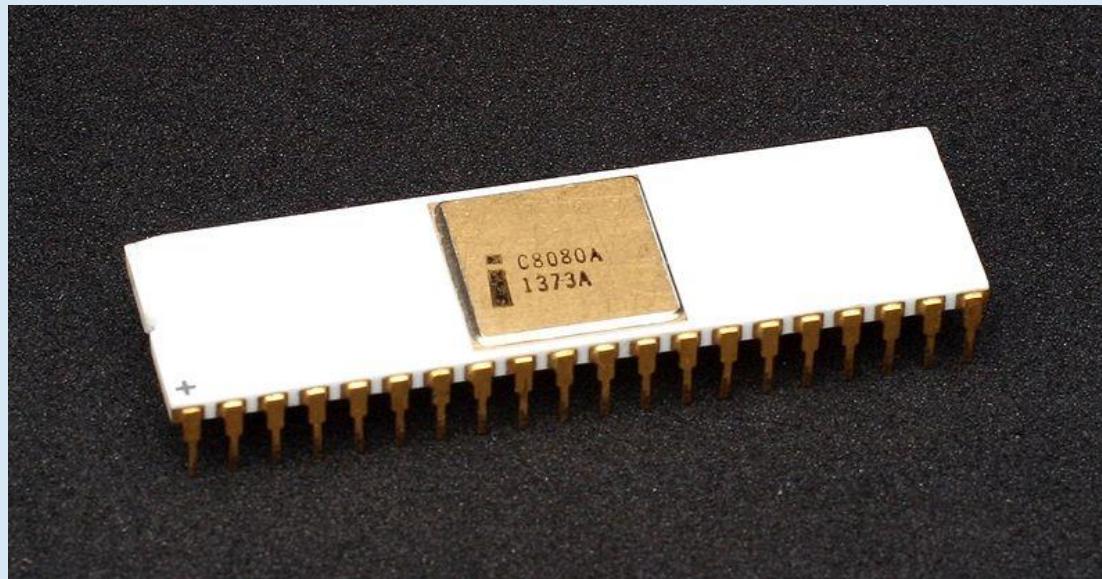
- **Evolução dos microprocessadores Intel:**
- **Pentium II:** o Pentium II incorporou a **tecnologia Intel MMX**, que foi projetada especificamente para **processar dados de vídeo, áudio e gráfico de forma eficiente**.
- **Pentium III:** o Pentium III incorpora instruções de ponto flutuante adicionais: a extensão de conjunto de instrução **The Streaming SIMD Extensions (SSS)** adicionou 70 novas instruções projetadas para aumentar o desempenho quando exatamente as mesmas operações estiverem para ser executadas **em objetos de dados múltiplos**. O **processamento digital e o processamento gráfico** são aplicações típicas.

A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **Pentium 4:** o Pentium 4 inclui ponto flutuante adicional e outras melhorias para multimídia.
- **Core:** este é o primeiro processador Intel x86 com **dual core**, referindo-se à implementação de dois cores em um único chip.
- **Core 2:** o Core 2 estende a arquitetura para 64 bits. O core 2 Quad oferece quatro processadores em um único chip. As ofertas mais recentes de Core têm até 10 cores por chip. O conjunto de instrução Advanced Vector Extensions foi um importante incremento, que ofereceu um conjunto de instruções para processamento eficiente de vetores de dados de 256 bits, e, então de 512 bits.

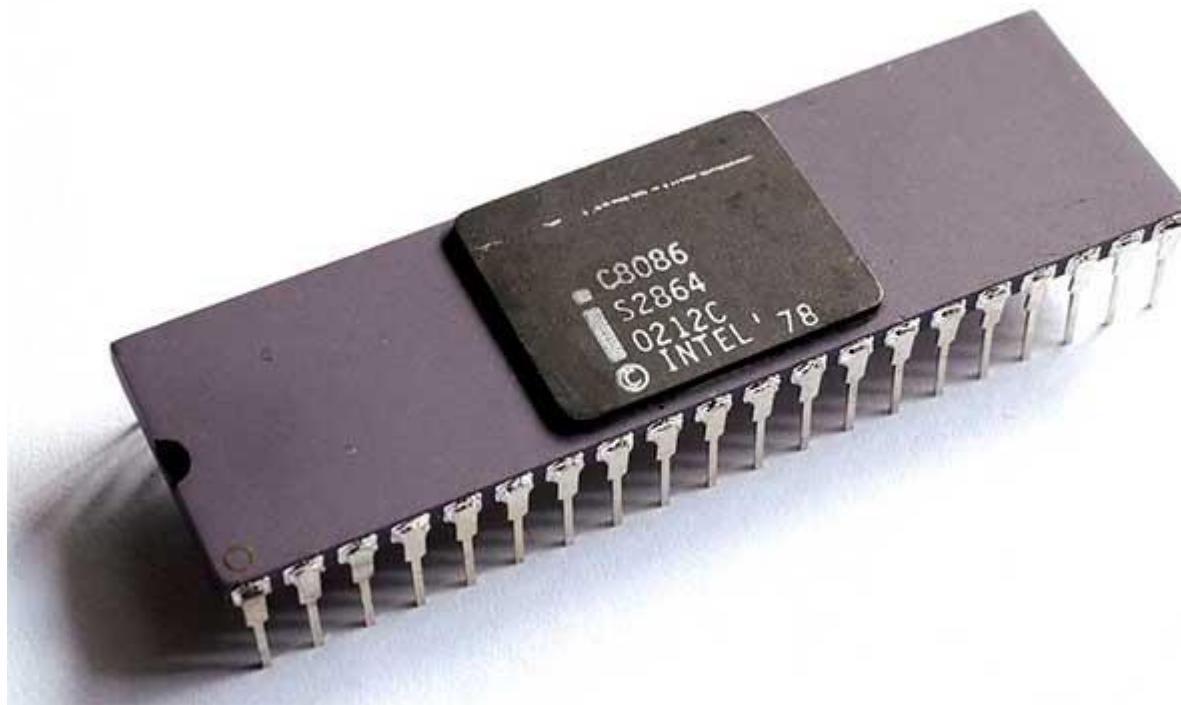
A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **Processador 8080**



A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **Processador 8086**



A evolução da arquitetura Intel x86

- Evolução dos microprocessadores Intel:
- Processador 80286



A evolução da arquitetura Intel x86

- Evolução dos microprocessadores Intel:
- Processador 80386



A evolução da arquitetura Intel x86

- Evolução dos microprocessadores Intel:
- Processador 80486



A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **Processador Pentium**



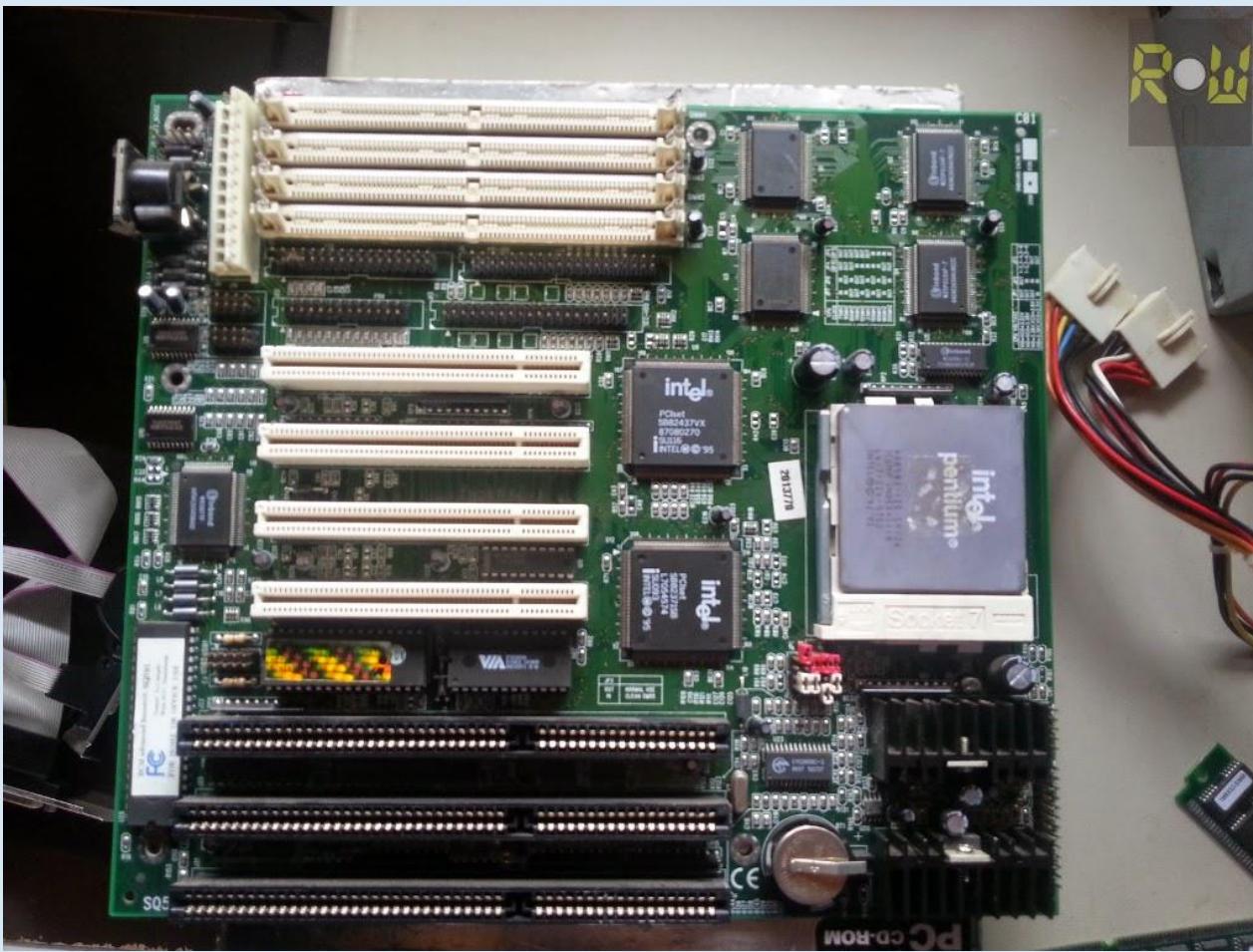
A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **Processador Pentium Pro**



A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **Processador Pentium II**



A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **Processador Pentium III**



A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **Processador Pentium 4**



A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **Processador Core**



A evolução da arquitetura Intel x86

- **Evolução dos microprocessadores Intel:**
- **Processador Core 2 Quad**



A evolução da arquitetura Intel x86

➤ Evolução dos microprocessadores Intel:

(a) Processadores da década de 1970					
	4004	8008	8080	8086	8088
Introduzido	1971	1972	1974	1978	1979
Velocidade de clock	108 kHz	108 kHz	2 MHz	5 MHz, 8 MHz, 10 MHz	5 MHz, 8 MHz
Largura do barramento	4 bits	8 bits	8 bits	16 bits	8 bits
Número de transistores	2.300	3.500	6.000	29.000	29.000
Dimensão da tecnologia de fabricação (μm)	10	8	6	3	6
Memória endereçável	640 bytes	16 kB	64 kB	1 MB	1 MB

A evolução da arquitetura Intel x86

➤ Evolução dos microprocessadores Intel:

	(b) Processadores da década de 1980			
	80286	386TM DX	386TM SX	486TM DX CPU
Introduzido	1982	1985	1988	1989
Velocidade de clock	6–12,5 MHz	16–33 MHz	16–33 MHz	25–50 MHz
Largura do barramento	16 bits	32 bits	16 bits	32 bits
Número de transistores	134.000	275.000	275.000	1,2 milhão
Dimensão da tecnologia de fabricação (μm)	1,5	1	1	0,8–1
Memória endereçável	16 MB	4 GB	16 MB	4 GB
Memória virtual	1 GB	64 TB	64 TB	64 TB
Cache	—	—	—	8 kB

A evolução da arquitetura Intel x86

➤ Evolução dos microprocessadores Intel:

(c) Processadores da década de 1990				
	486TM SX	Pentium	Pentium Pro	Pentium II
Introduzido	1991	1993	1995	1997
Velocidade de clock	16–33 MHz	60–166 MHz,	150–200 MHz	200–300 MHz
Largura do barramento	32 bits	32 bits	64 bits	64 bits
Número de transistores	1,185 milhão	3,1 milhões	5,5 milhões	7,5 milhões
Dimensão da tecnologia de fabricação (μm)	1	0,8	0,6	0,35
Memória endereçável	4 GB	4 GB	64 GB	64 GB
Memória virtual	64 TB	64 TB	64 TB	64 TB
Cache	8 kB	8 kB	512 kB L1 e 1 MB L2	512 kB L2

A evolução da arquitetura Intel x86

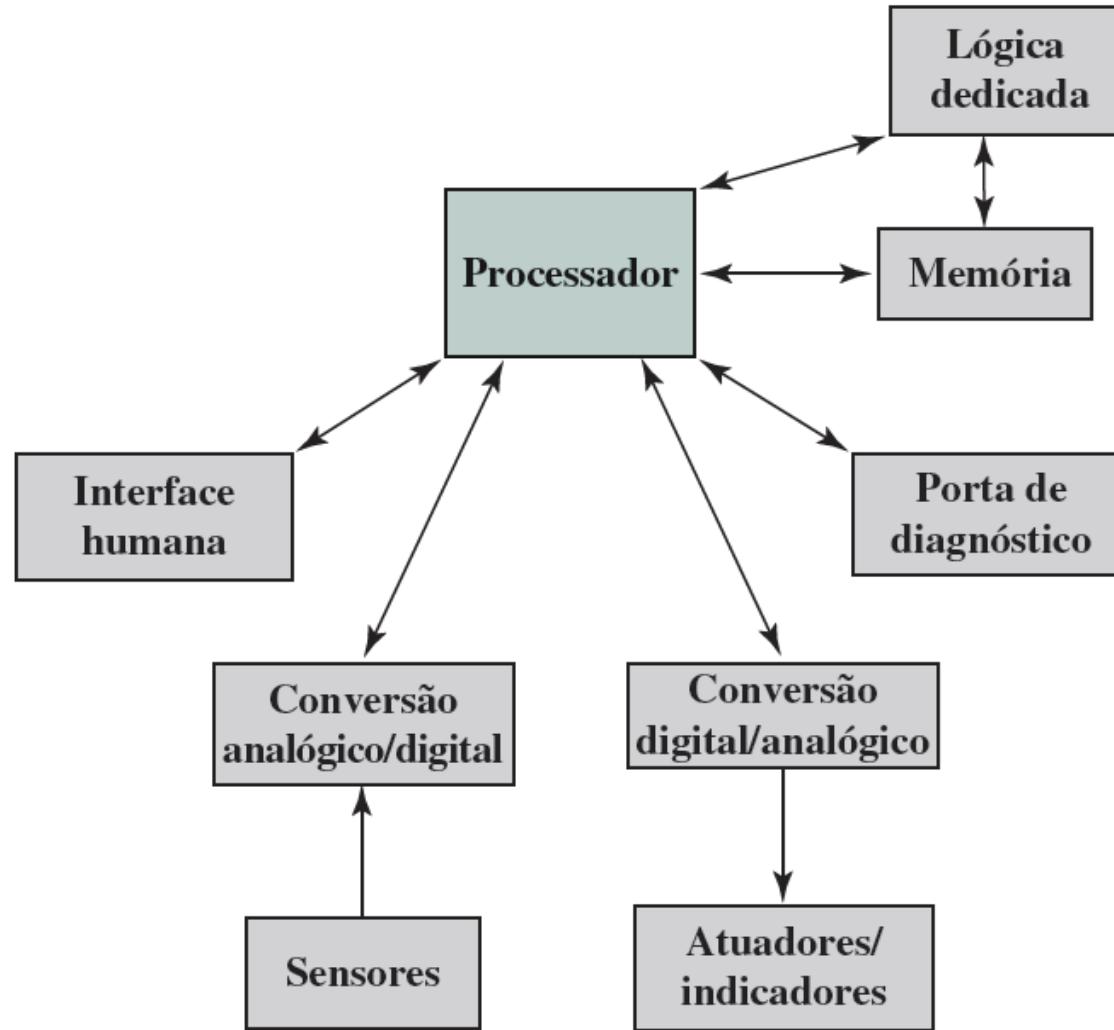
➤ Evolução dos microprocessadores Intel:

	(d) Processadores recentes			
	Pentium III	Pentium 4	Core 2 Duo	Core i7 EE 4960X
Introduzido	1999	2000	2006	2013
Velocidade de clock	450–660 MHz	1,3–1,8 GHz	1,06–1,2 GHz	4 GHz
Largura do barramento	64 bits	64 bits	64 bits	64 bits
Número de transistores	9,5 milhões	42 milhões	167 milhões	1,86 bilhão
Dimensão da tecnologia de fabricação (nm)	250	180	65	22
Memória endereçável	64 GB	64 GB	64 GB	64 GB
Memória virtual	64 TB	64 TB	64 TB	64 TB
Cache	512 kB L2	256 kB L2	2 MB L2	1,5 MB L2/15 MB L3
Número de cores	1	1	2	6

Sistemas Embarcados

- O termo **sistema embarcado** refere-se **ao uso de eletrônica e software dentro de um produto**, ao contrário de um computador de uso geral, como um sistema de laptop ou desktop.
- Hoje em dia, alguns, ou a maioria, dos dispositivos que usam energia elétrica têm um sistema computacional embarcado.
- A figura a seguir mostra em termos gerais uma organização de sistema embarcado.
- Além do processador e da memória, existem diversos elementos que diferem do desktop ou laptop típico.

Sistemas Embarcados



A Internet das Coisas

- A **Internet das Coisas** (IoT — do inglês, *Internet of Things*) é um termo que se refere à interconexão expansiva dos dispositivos inteligentes, indo de aplicações a minúsculos sensores.
- Com referência aos sistemas terminais suportados, a internet passou por cerca de **quatro gerações de implantação**:
 1. **Tecnologia da informação (TI)**
 2. **Tecnologia operacional (TO)**
 3. **Tecnologia pessoal**
 4. **Tecnologia de sensor/atuador**

A Internet das Coisas

- A **Internet das Coisas** (IoT — do inglês, *Internet of Things*) é um termo que se refere à interconexão expansiva dos dispositivos inteligentes, indo de aplicações a minúsculos sensores.
- Com referência aos sistemas terminais suportados, a internet passou por cerca de **quatro gerações de implantação**:
 1. **Tecnologia da informação (TI)**: PCs, servidores, roteadores, firewalls, e assim por diante, comprados como dispositivos por profissionais de TI e que usam principalmente conectividade com fios.
 2. **Tecnologia operacional (TO)**: equipamentos/aplicações com TI embarcada criados por empresas que não são de TI, como equipamentos médico, SCADA (controle de supervisão e aquisição de dados, do inglês *Supervisory Control And Data Acquisition*), controle de processo e quiosques, comprados como aplicações por profissionais de empresas de TO, e que usam principalmente conectividade com fios.

A Internet das Coisas

- A **Internet das Coisas** (IoT — do inglês, *Internet of Things*) é um termo que se refere à interconexão expansiva dos dispositivos inteligentes, indo de aplicações a minúsculos sensores.
- Com referência aos sistemas terminais suportados, a internet passou por cerca de **quatro gerações de implantação**:
 3. **Tecnologia pessoal:** smartphones, tablets e leitores de e-books comprados como dispositivo se TI por consumidores que usam exclusivamente conectividade sem fio e diversas outras formas deste tipo de conectividade.
 4. **Tecnologia de sensor/atuador:** dispositivos de uso único comprados por consumidores, pessoas de TI e TO que usam exclusivamente conectividade sem fio, geralmente de forma simples, como parte de sistemas maiores.

Sistemas operacionais embarcados

- Há duas técnicas gerais para desenvolver o sistema operacional (SO) embarcado:
 1. A primeira técnica é pegar um SO existente e adaptar para a aplicação embarcada. Por exemplo, há versões embarcadas de Linux, Windows e Mac, bem como outros sistemas operacionais comerciais e particulares especializados para sistemas embarcados.
 2. A outra técnica é desenvolver e implementar um SO direcionado unicamente para o uso embarcado. Um exemplo é o TinyOS, amplamente usado em redes de sensor sem fio.

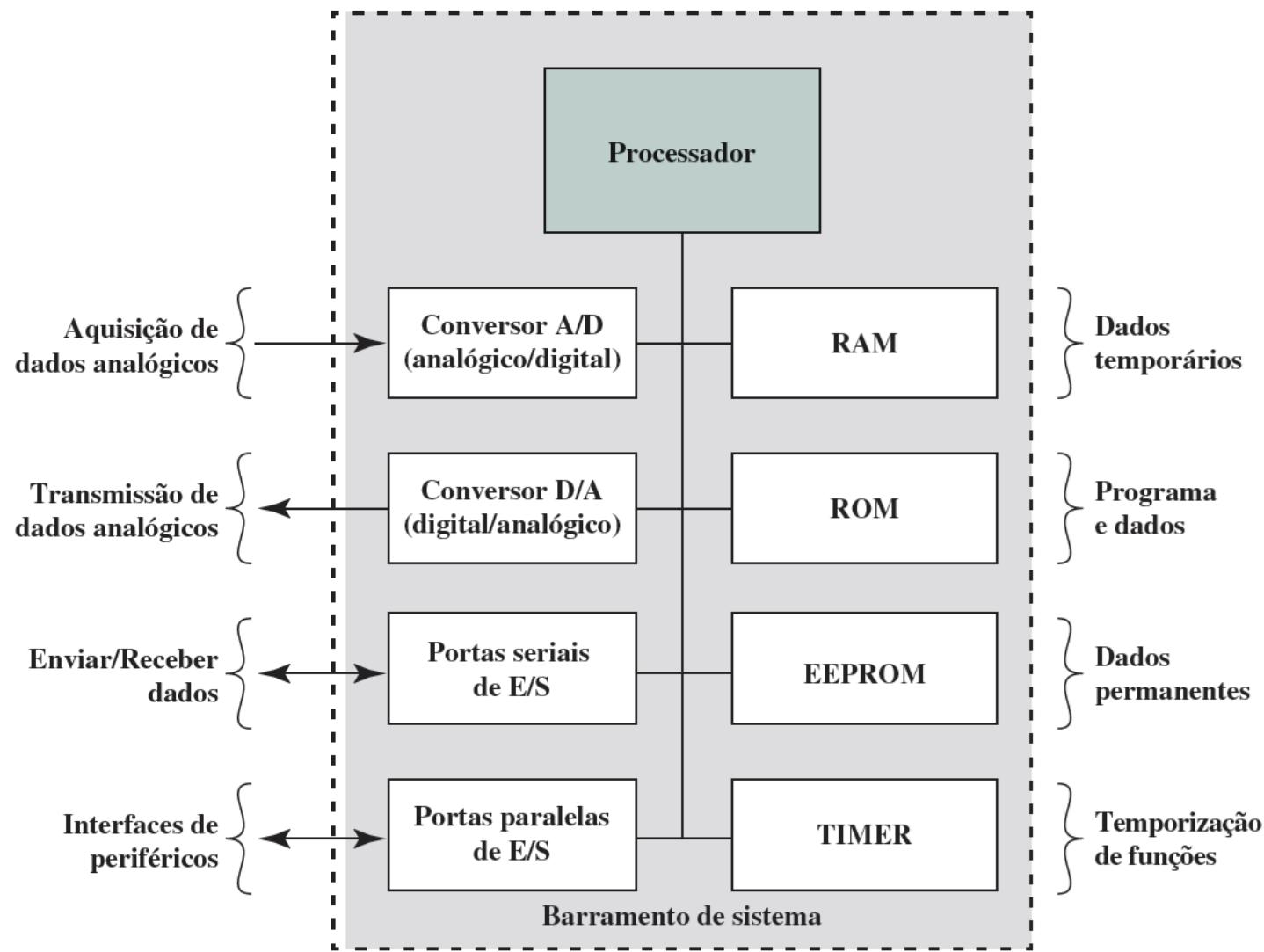
Processadores para aplicações versus processadores dedicados

- **Processadores de aplicações** são definidos pela capacidade do processador de executar sistemas operacionais complexos, como Linux, Android e Chrome.
- O processador de aplicações é **naturalmente de uso geral**.
- Um **processador dedicado** é dedicado a uma ou a algumas poucas tarefas específicas exigidas pelo dispositivo hospedeiro.
- Por conta de tal **sistema embarcado ser dedicado a uma tarefa ou a tarefas específicas**, o processador e os componentes associados podem ser construídos para reduzir o tamanho e o custo.

Microprocessadores versus microcontroladores

- Os primeiros **chips de microprocessadores** incluíam registradores, uma ALU e algum tipo de unidade de controle ou de lógica de processamento de instrução.
- Chips de microprocessadores atuais incluem diversos cores e uma quantidade substancial de memória cache.
- Um **chip microcontrolador** faz uso substancialmente diferente do espaço de lógica.
- A figura a seguir mostra em termos gerais os elementos tipicamente encontrados em um chip microcontrolador.

Microprocessadores versus microcontroladores



Arquitetura ARM

- A arquitetura ARM refere-se a uma arquitetura de processador que evoluiu dos princípios de desenvolvimento do RISC e é usada em sistemas embarcados.
- Os chips ARM são processadores de alta velocidade que são conhecidos pelo pequeno tamanho do die e pelo baixo consumo de energia.
- Os chips ARM são os processadores dos populares dispositivos Apple, o iPod e o iPhone, e são usados em praticamente todos os smartphones Android.
- O conjunto de instruções ARM é altamente regular.

Arquitetura ARM

- A [ARM Holdings](#) licencia um número de microprocessadores especializados e relacionados às tecnologias, mas a maior parte de sua linha de produtos é a família das arquiteturas de microprocessadores Cortex.
- Há três arquiteturas Cortex, convenientemente denominadas pelas iniciais A, R e M:
 1. Cortex-A e Cortex-A50
 2. Cortex-R
 3. Cortex-M

Computação em nuvem

- A NIST(*National Institute of Standards and Technology*) define a computação em nuvem, em NIST SP-800-145:
- **Computação em nuvem**
 - Um modelo para possibilitar acesso onipresente, conveniente e sob demanda a um grupo compartilhado de recursos de computação configuráveis que pode ser rapidamente fornecido e liberado com um esforço mínimo de gerenciamento ou interação do provedor de serviço.
 - A **rede em nuvem** refere-se às redes e funcionalidades de gerenciamento de rede que devem estar em ordem para possibilitar a computação em nuvem.

Computação em nuvem

- O **armazenamento em nuvem** consiste em um armazenamento de base de dados e aplicações de base de dados hospedadas nos servidores da nuvem.
- O **provedor de serviço de nuvem** (CSP — do inglês, *Cloud Service Provider*) mantém os recursos de computação e armazenamento de dados que estão disponíveis na internet ou em redes privadas.
- Praticamente todos os serviços de nuvem são provados pelo uso de um dos três modelos (apresentados na figura a seguir): SaaS, PaaS e IaaS.

Computação em nuvem

