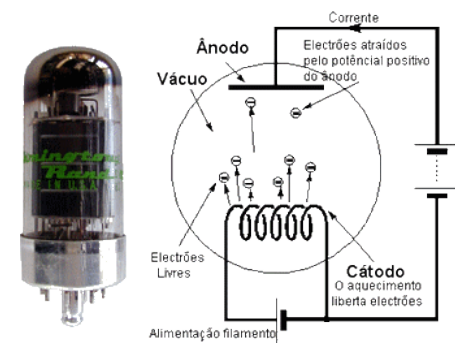
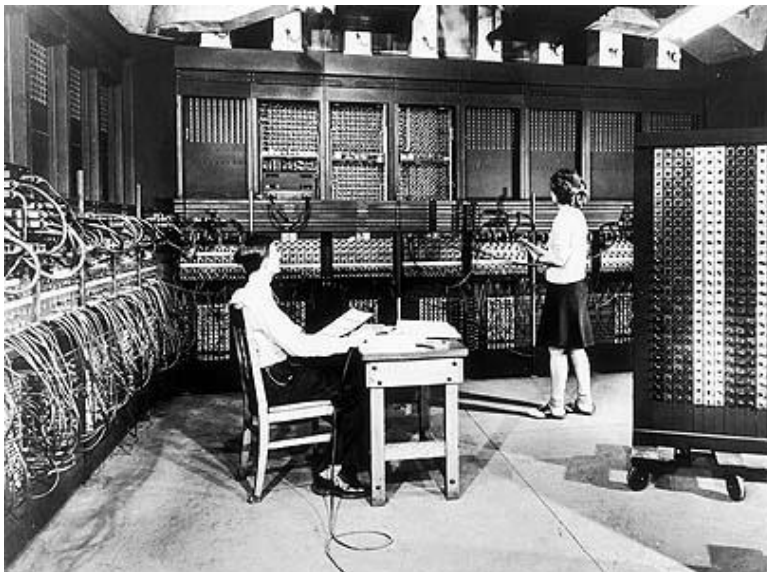
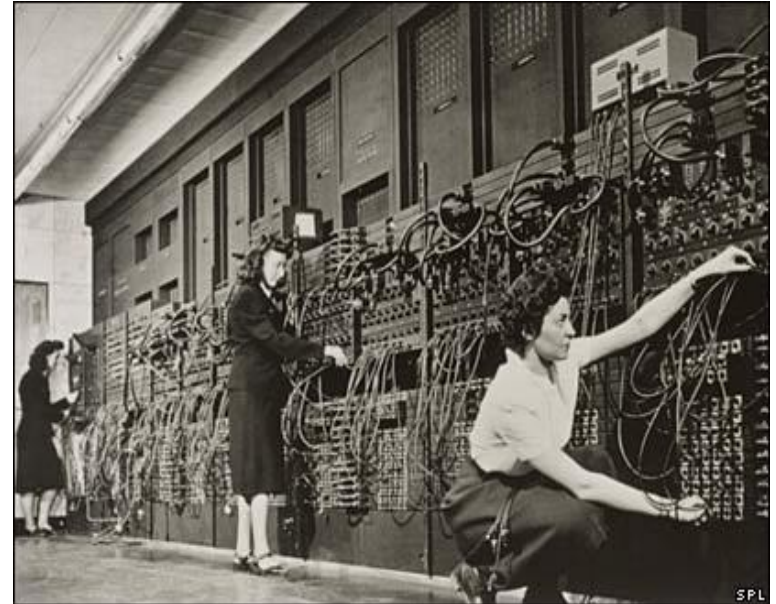
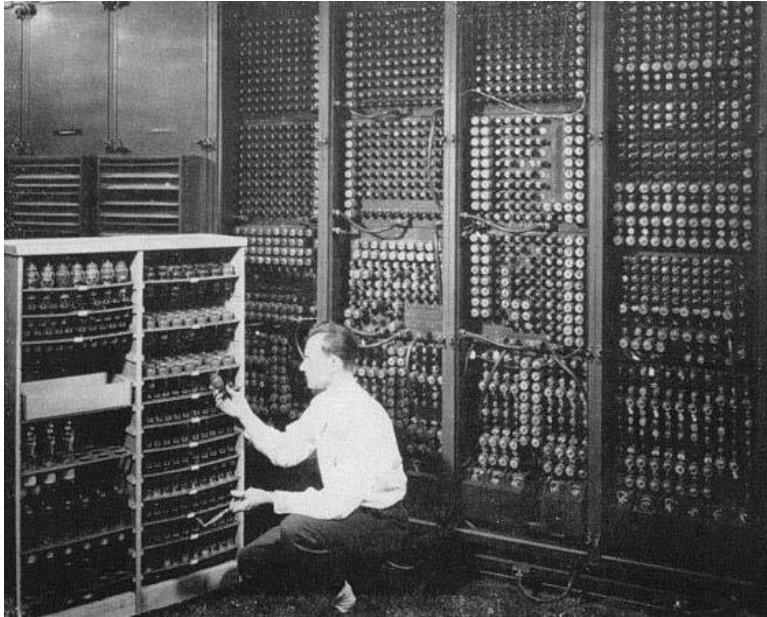


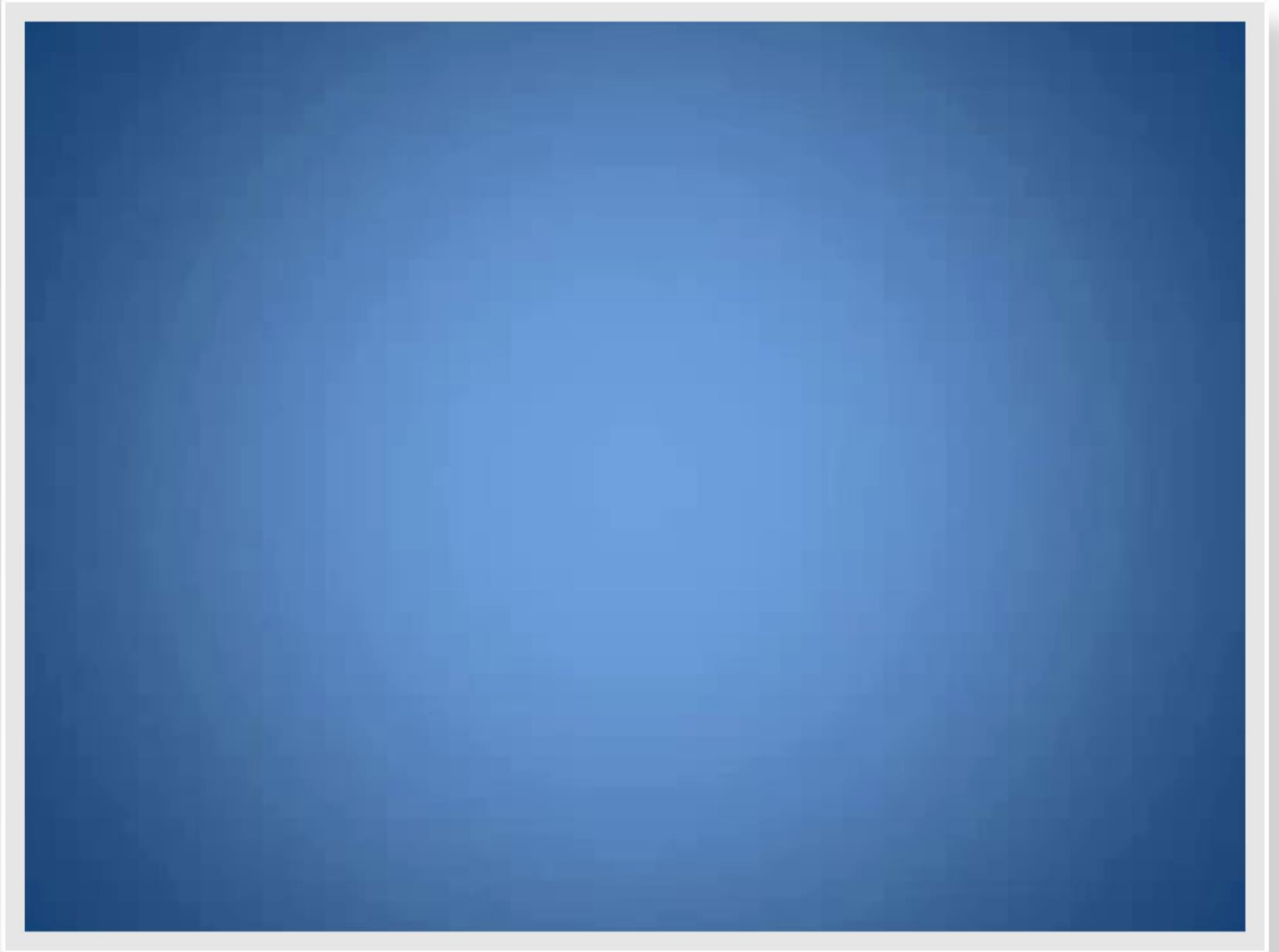
# ENIAC (Electronic Numerical Integrator And Computer) – histórico e detalhes

- Primeiro computador digital eletrônico de uso geral do mundo.
- 30 toneladas, ocupava uma área de 457,2 m<sup>2</sup> 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.

## ENIAC – Fotos



## ENIAC – Vídeo

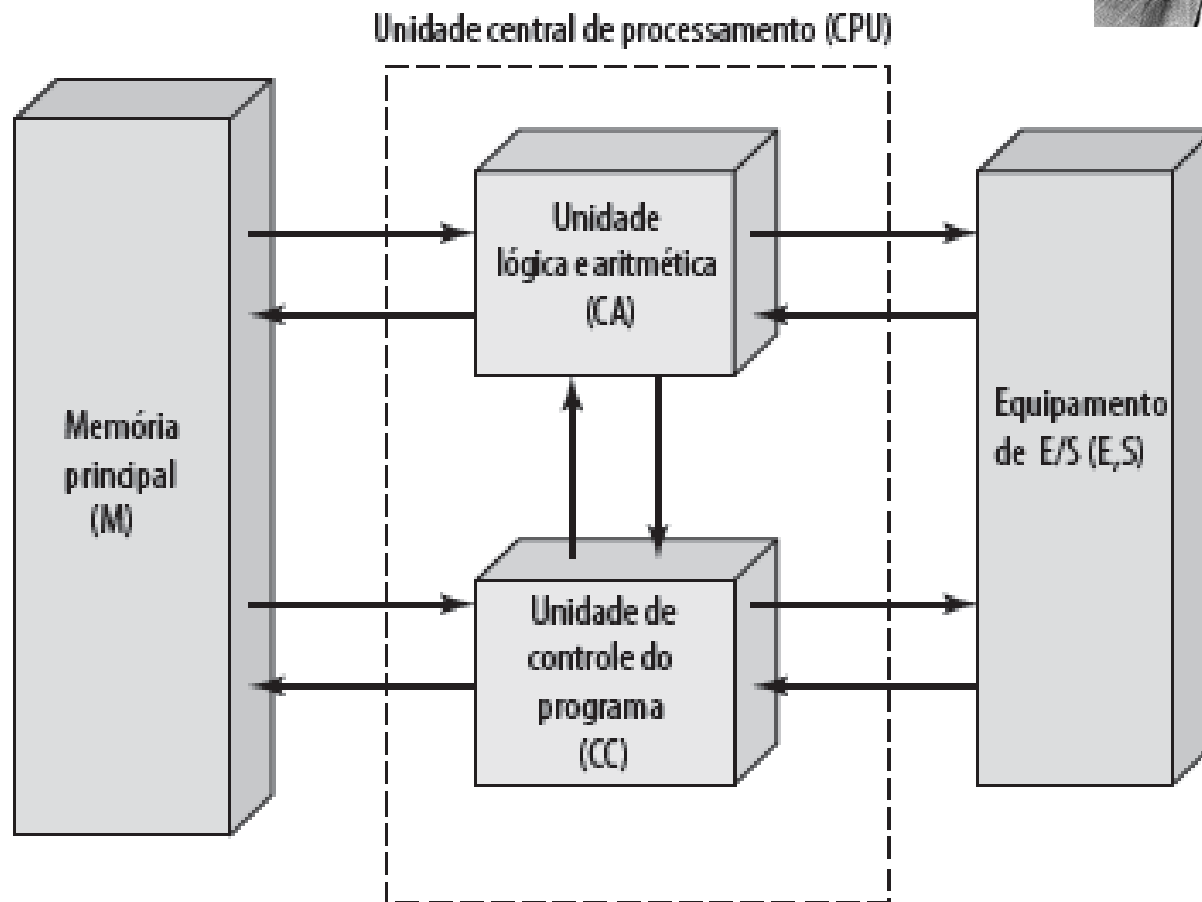


## Von Neumann/Turing

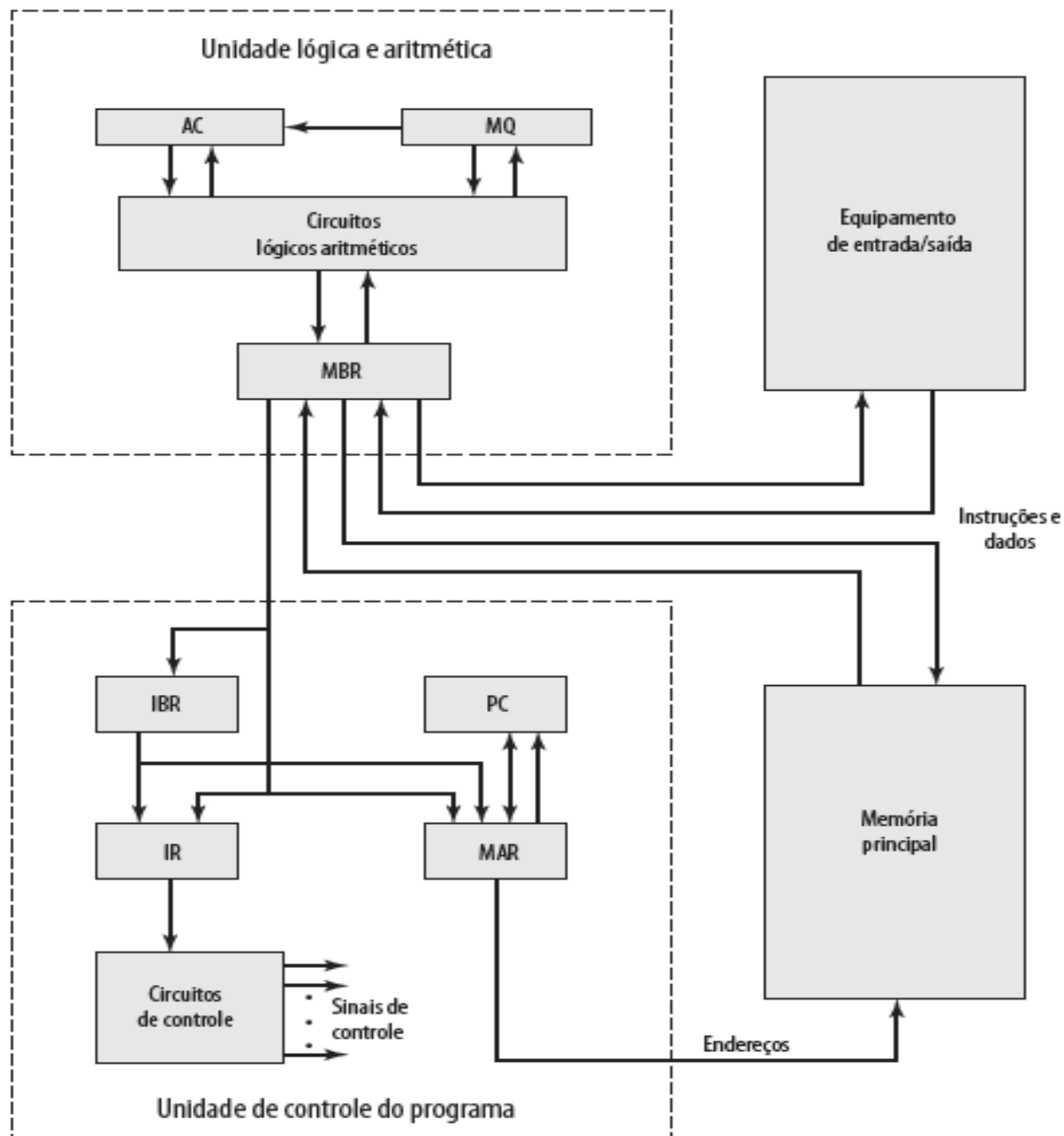


- John von Neumann/Alan Turing
- Conceito de programa armazenado.
  - Princeton Institute for Advanced Studies - IAS
- Estrutura geral do IAS (concluído em 1952)
  - Memória principal armazenando instruções e dados.
  - ALU operando sobre dados binários.
  - Unidade de controle interpretando e executando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

## Estrutura da máquina de Von Neumann



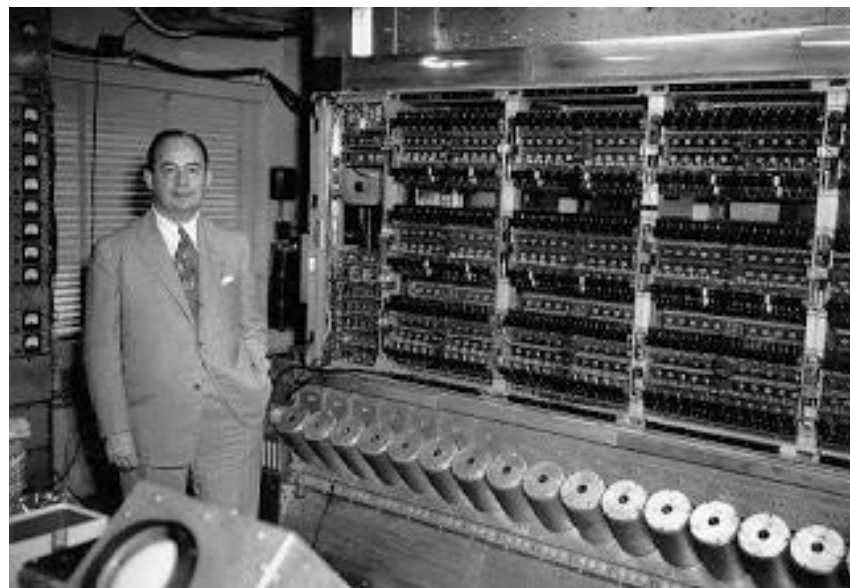
## Estrutura do IAS – detalhe



# IAS – “Conceito de Programa Armazenado”

## 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 Address Register)**: especifica o endereço na memória da palavra a ser escrita ou lida no MBR).





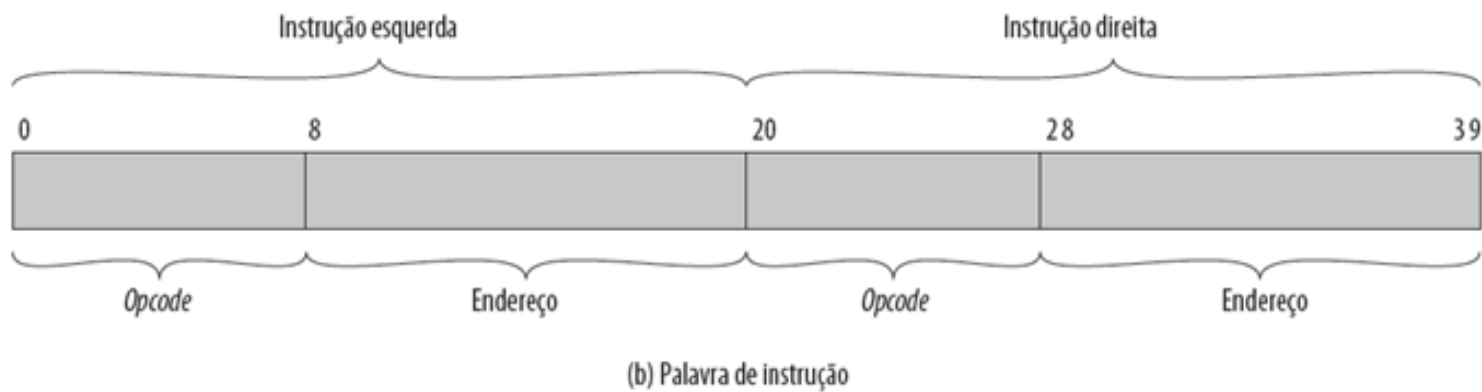
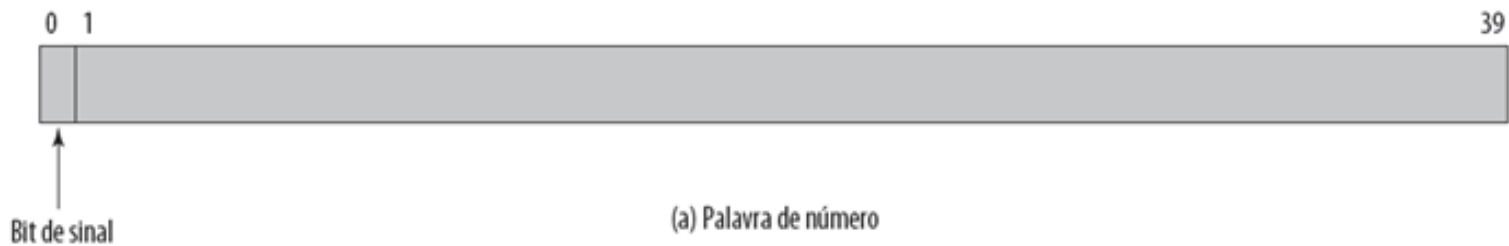
## IAS – detalhes

- **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.



## IAS – detalhes

### Formato de memória



## 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.

## 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)	$AC \leftarrow -\text{abs}(\text{mem}(X))$

## 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)

## IAS – detalhes

## Agrupamento das instruções

## Instruções de desvio condicional

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

## IAS – detalhes

## Agrupamento das instruções

## Instruções aritméticas

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

## IAS – detalhes

## Agrupamento das instruções

## Instruções de alteração de endereço

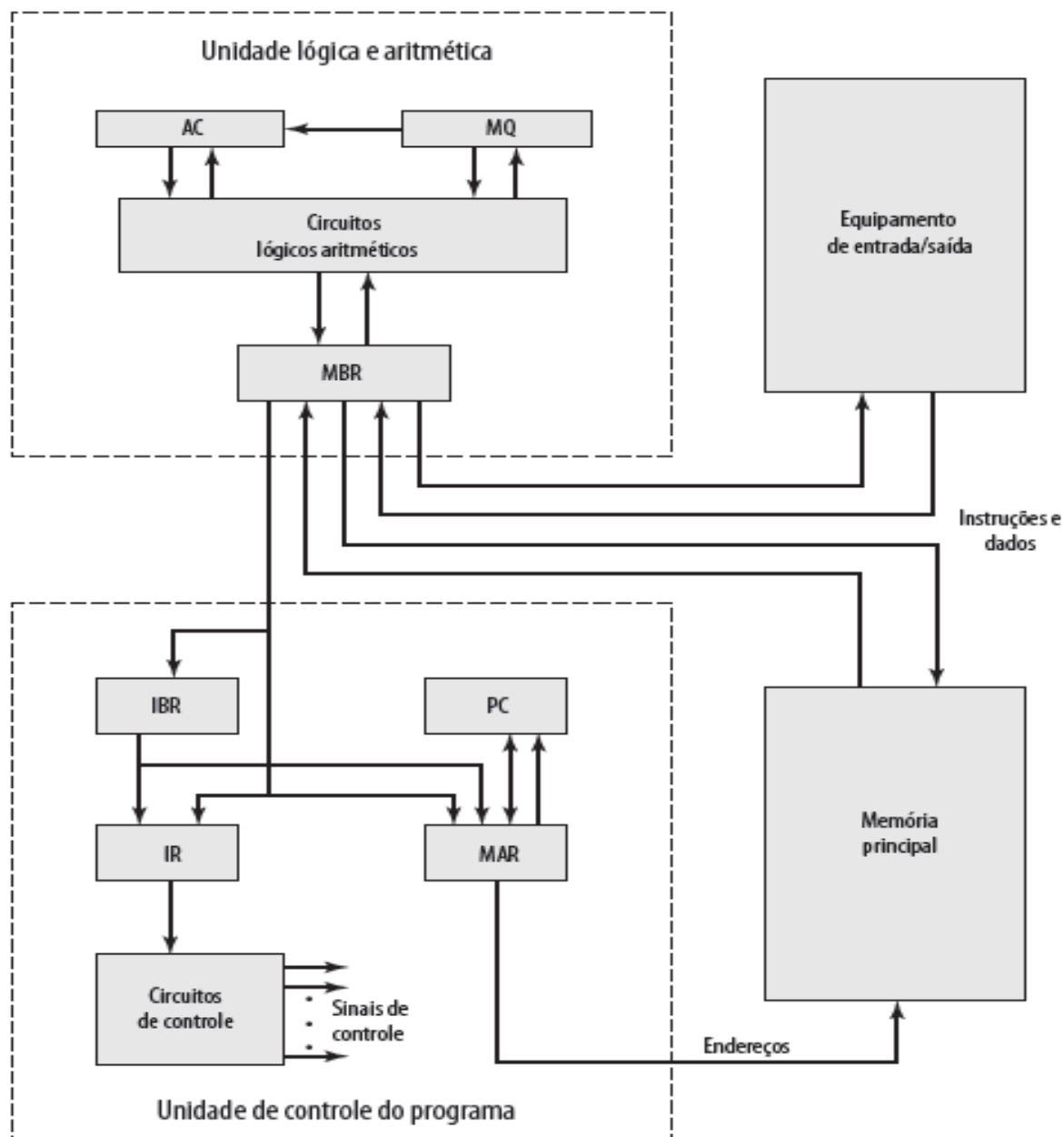
opcode	mnemônico	significado
00010010	STOR M(X,8:19)	substitui o campo de endereço à esq. de mem(X) pelos 12 bits mais à dir. de AC
00010011	STOR M(X,28:39)	subst. o campo de endereço à dir. de mem(X) pelos 12 bits mais à esq. de AC



## Estrutura do IAS – detalhe – Conj. de instruções do IAS

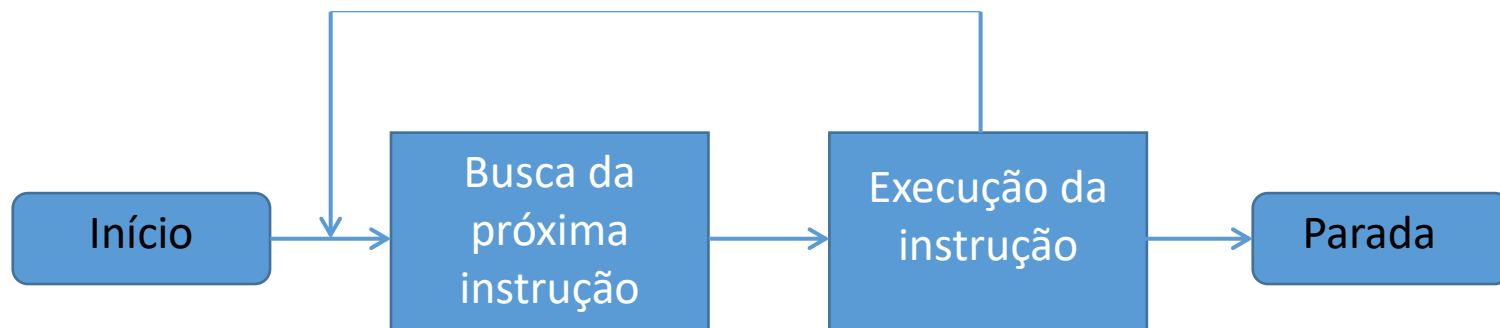
Tipo de instrução	Opcode	Representação simbólica	Descrição
Transferência de dados	00001010	LOAD MQ	Transfere o conteúdo de MQ para AC
	00001001	LOAD MQ,M(X)	Transfere o conteúdo do local de memória X para MQ
	00100001	STOR M(X)	Transfere o conteúdo de AC para o local de memória X
	00000001	LOAD M(X)	Transfere M(X) para o AC
	00000010	LOAD – M(X)	Transfere – M(X) para o AC
	00000011	LOAD  M(X)	Transfere o valor absoluto de M(X) para o AC
	00000100	LOAD –  M(X)	Transfere - M(X)  para o acumulador
Desvio incondicional	00001101	JUMP M(X,0:19)	Apanha a próxima instrução da metade esquerda de M(X)
	00001110	JUMP M(X,20:39)	Apanha a próxima instrução da metade direita de M(X)
Desvio condicional	00001111	JUMP+ M(X,0:19)	Se o número no AC for não negativo, apanha a próxima instrução da metade esquerda de M(X)
	00010000	JUMP+ M(X,20:39)	Se o número no AC for não negativo, apanha a próxima instrução da metade direita de M(X)
Aritmética	00000101	ADD M(X)	Soma M(X) a AC; coloca o resultado em AC
	00000111	ADD  M(X)	Soma  M(X)  a AC; coloca o resultado em AC
	00000110	SUB M(X)	Subtrai M(X) de AC; coloca o resultado em AC
	00001000	SUB  M(X)	Subtrai  M(X)  de AC; coloca o resto em AC
	00001011	MUL M(X)	Multiplica M(X) por MQ; coloca os bits mais significativos do resultado em AC; coloca bits menos significativos em MQ
	00001100	DIV M(X)	Divide AC por M(X); coloca o quociente em MQ e o resto em AC
	00010100	LSH	Multiplica o AC por 2; ou seja, desloca à esquerda uma posição de bit
	00010101	RSH	Divide o AC por 2; ou seja, desloca uma posição à direita
Modificação de endereço	00010010	STOR M(X,8:19)	Substitui campo de endereço da esquerda em M(X) por 12 bits mais à direita de AC
	00010011	STOR M(X,28:39)	Substitui campo de endereço da direita em M(X) por 12 bits mais à direita de AC

## Estrutura do IAS – detalhe



## Estrutura do IAS – detalhe

### Ciclo de vida de uma instrução



- **Busca:**

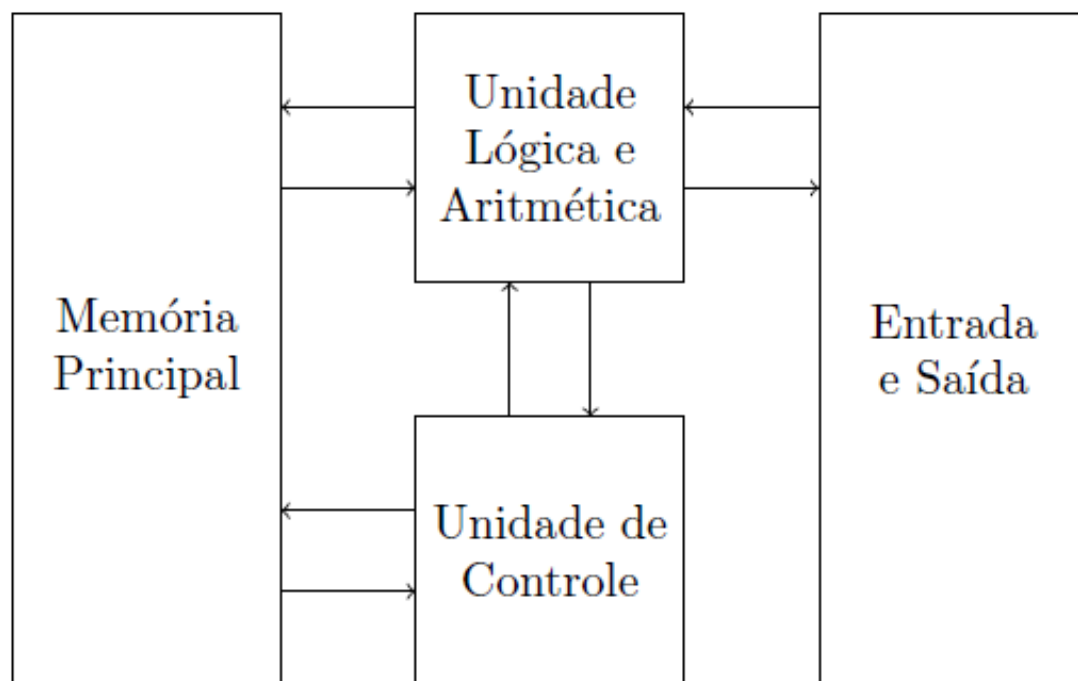
- Busca da instrução
- Atualiza PC
- Decodificação

- **Execução:**

- Busca operandos (se necessário)
- Executa
- Armazena resultado (se necessário)

## IAS

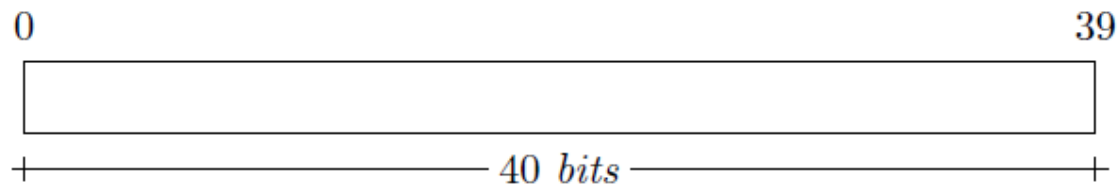
- Organização



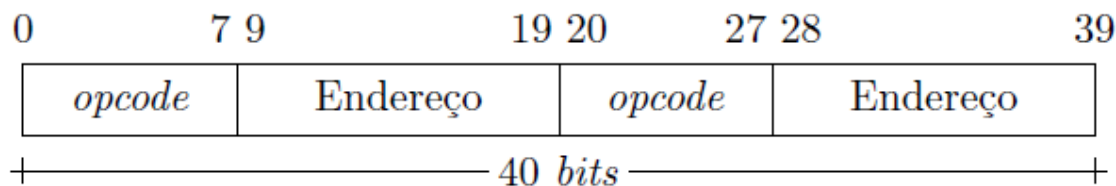
## IAS

- Organização da memória

	40 <i>bits</i>				
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



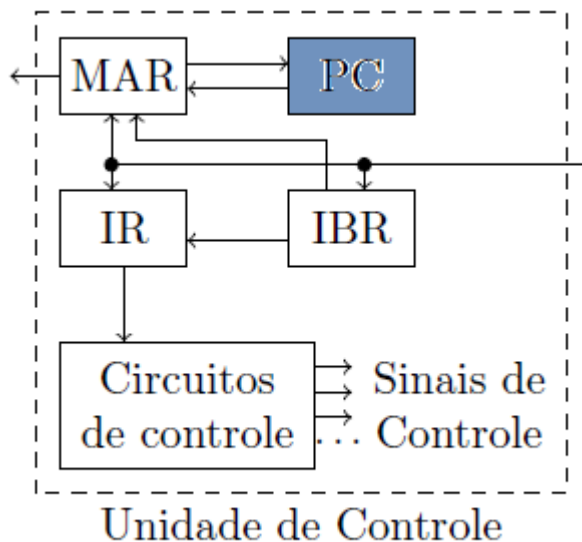
Um número de 40 *bits* em uma palavra da memória



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

# IAS

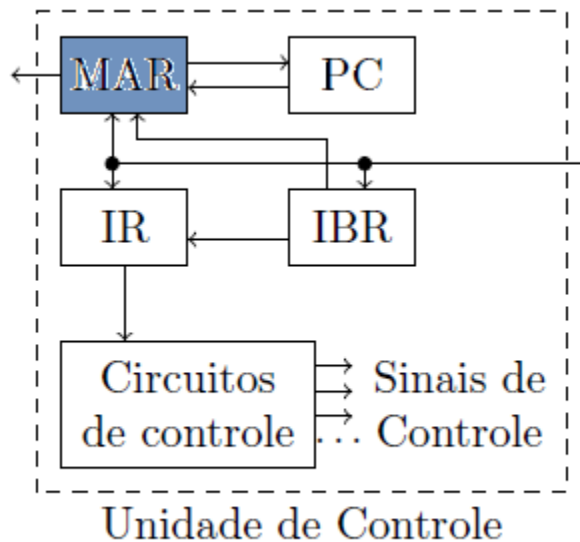
- Organização da Unidade de Controle (UC)



**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.**

## IAS

- Organização da Unidade de Controle (UC)

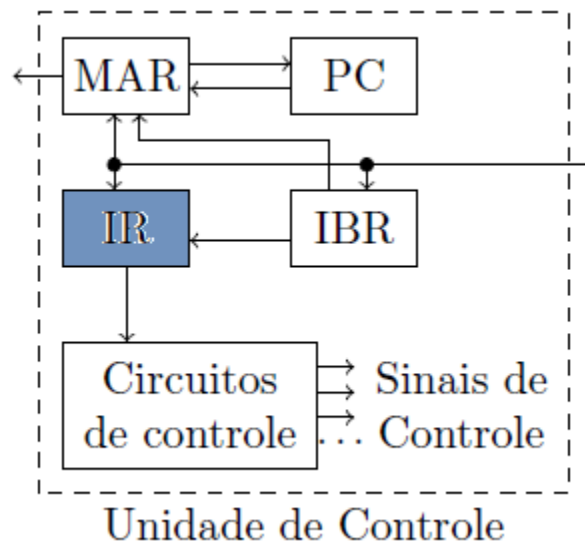


**MAR:** o *Memory Address Register*, ou registrador de endereço da memória, armazena um valor que representa um endereço de uma palavra da memória. **Este endereço será lido pela memória durante a operação de leitura ou escrita de dados.**



# IAS

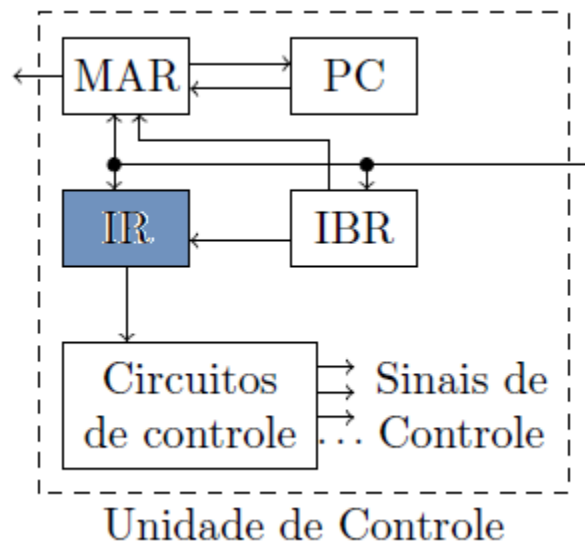
- Organização da Unidade de Controle (UC)



**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.

# IAS

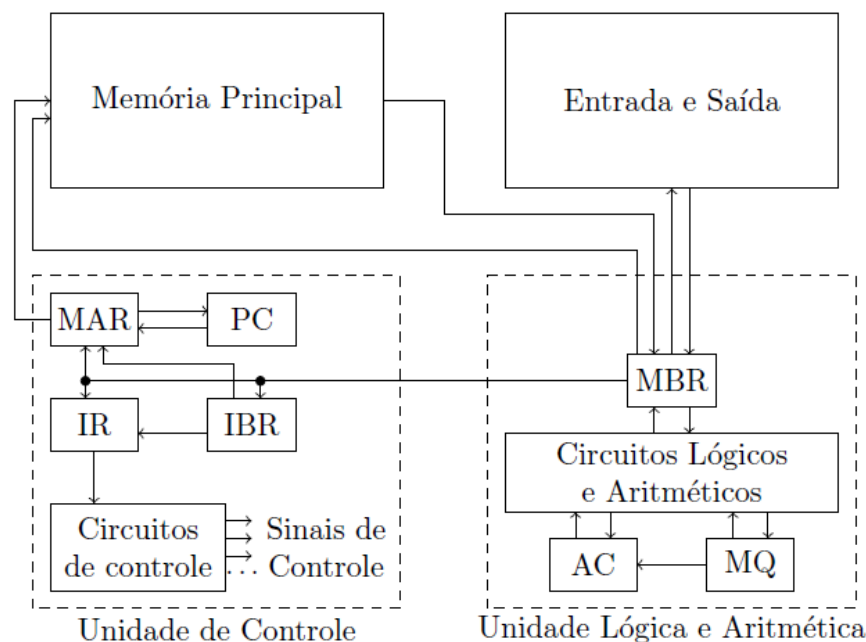
- Organização da Unidade de Controle (UC)



**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.

# IAS

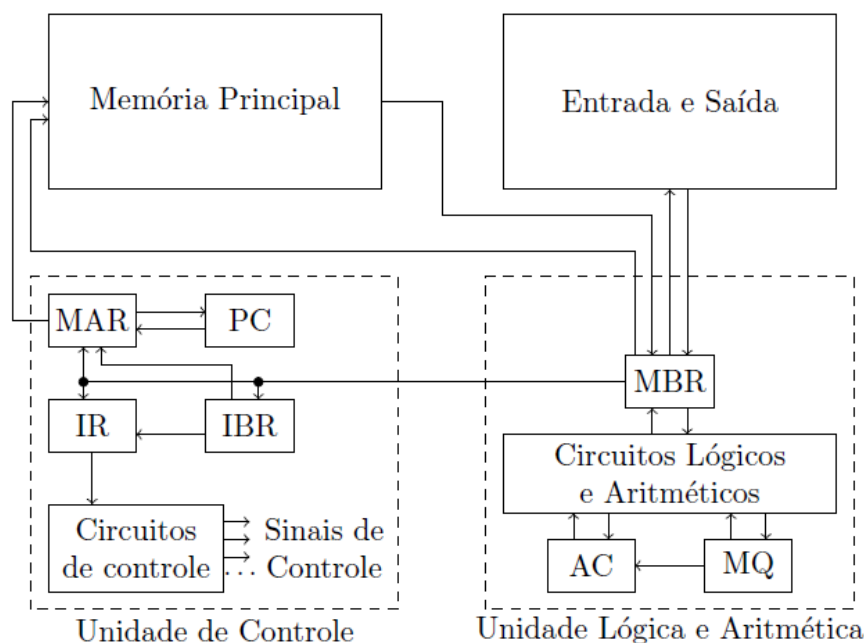
- Unidade Lógica e Aritmética (ULA)



**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.

# IAS

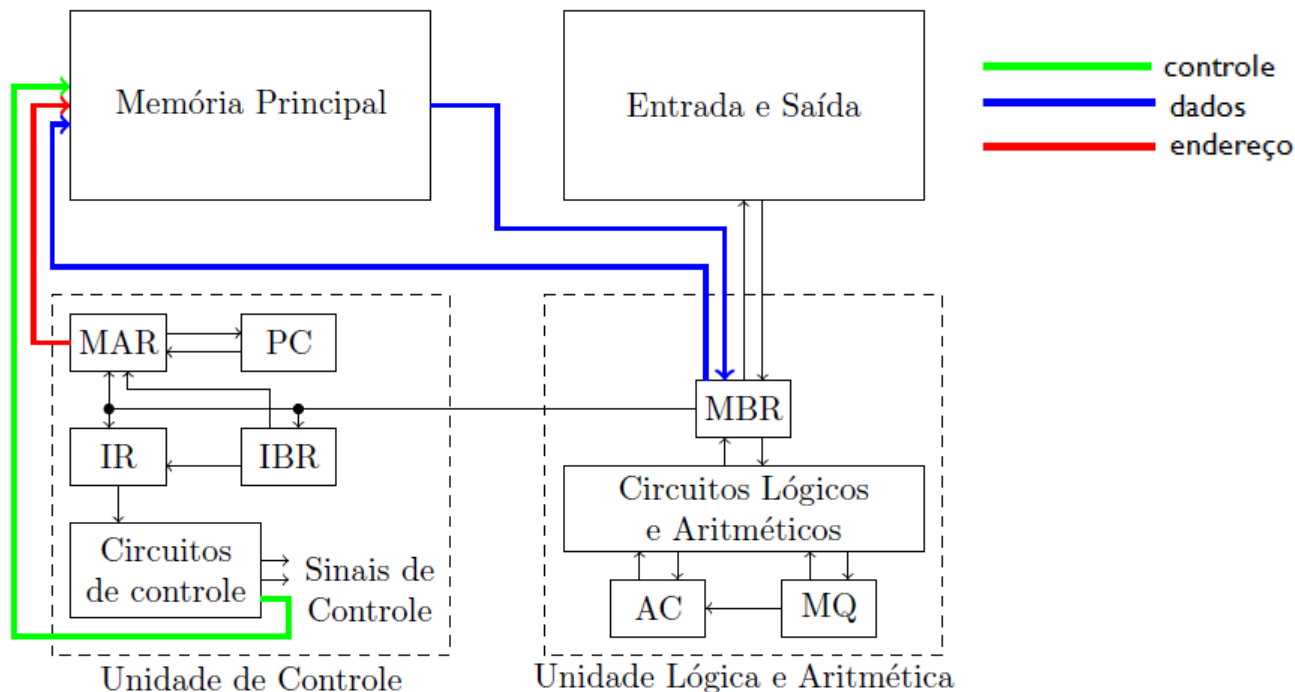
- Unidade Lógica e Aritmética (ULA)



**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.

# IAS

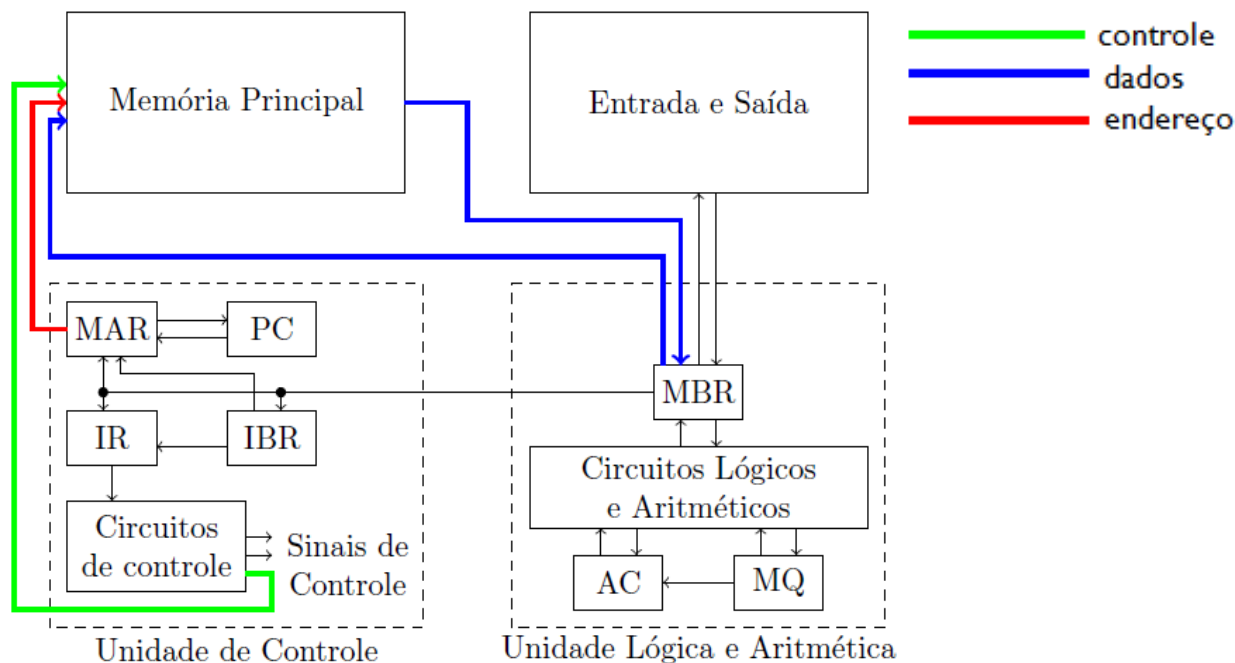
- Operação de leitura na memória



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.

# IAS

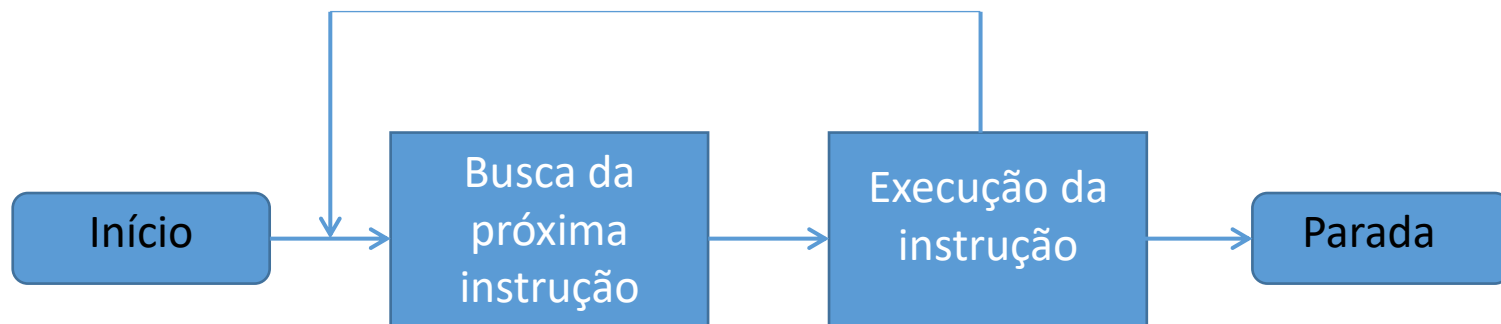
- Operação de escrita na memória



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;

# IAS

- Execução de instruções

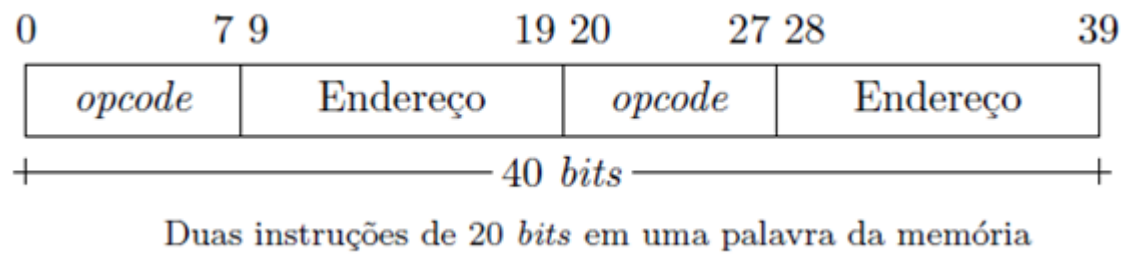


A execução de uma instrução é realizada em dois ciclos: o “ciclo de busca” e o “ciclo de execução”. O ciclo de busca consiste em buscar a instrução da memória (ou do registrador IBR) e armazenar no IR. O ciclo de execução, por sua vez, consiste em interpretar a instrução armazenada no registrador IR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



# IAS

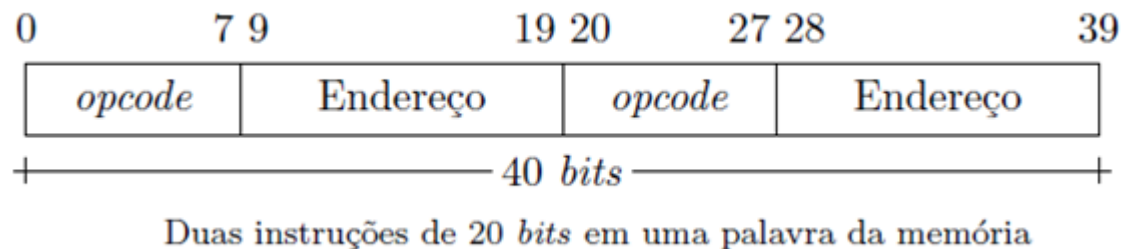
## • Ciclo de busca (à esquerda)



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  $PC + 1$ .

# IAS

- Ciclo de busca (à direita)

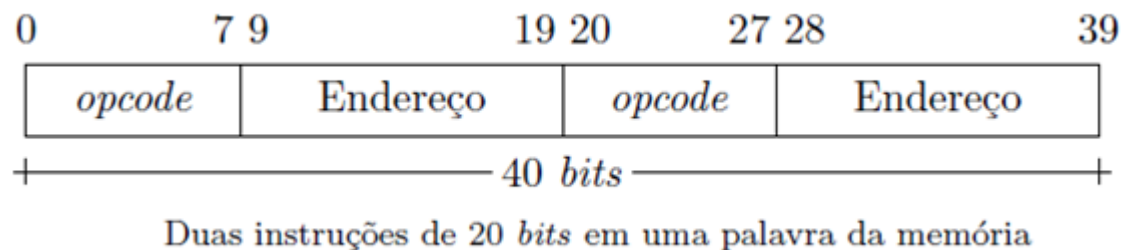


1. Se a última instrução executada foi a instrução à esquerda (**não houve desvio no fluxo de controle**), então:

- (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.

# IAS

- Ciclo de busca (à direita)

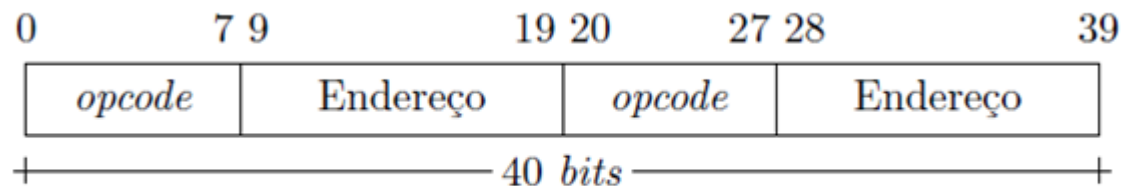


## 2. **Senão:**

- (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.

# IAS

- Ciclo de execução

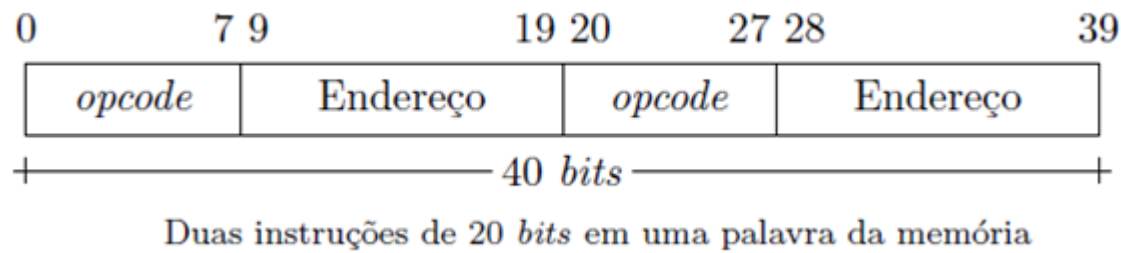


Duas instruções de 20 *bits* em uma palavra da memória

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;

# IAS

- Ciclo de execução

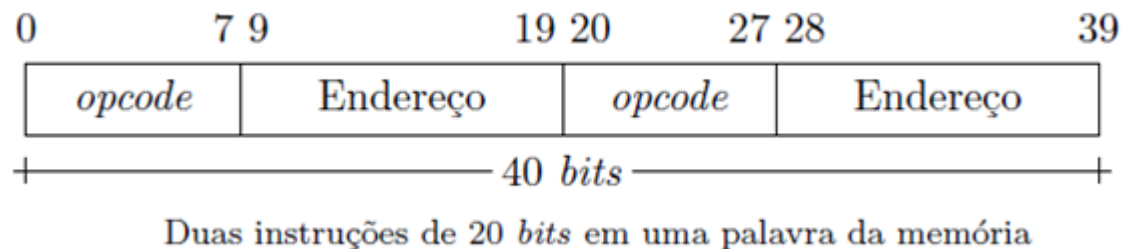


3. Se a instrução envolve a realização de uma operação lógica ou aritmética:

- (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.

# IAS

- Ciclo de execução

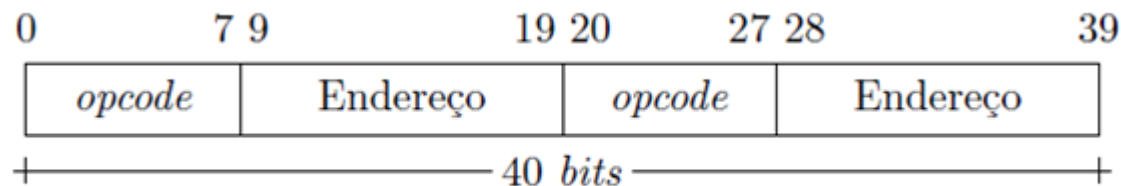


4. Se a instrução envolve a gravação do resultado na memória:

- (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.

# IAS

- Ciclo de execução



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

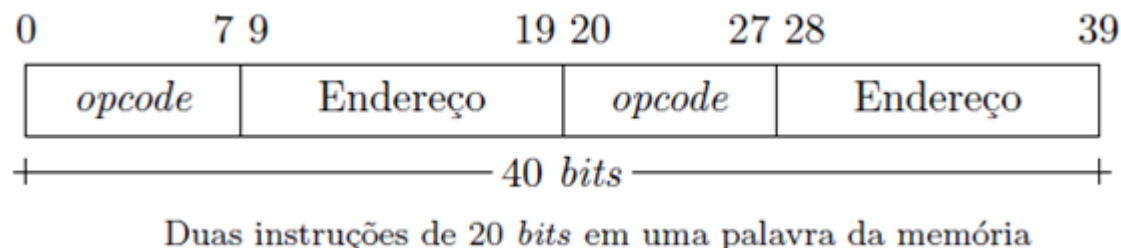
5. Se a execução da instrução implica no desvio do fluxo de controle, ou seja, se a instrução “salta” para uma outra instrução:

- (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.



# IAS

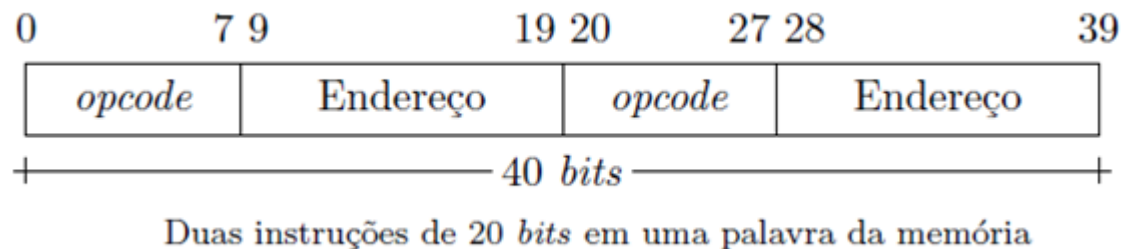
- Exemplo: ciclo de execução da instrução ADD(X)



- 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.
  - Após a identificação da instrução, o UC sabe que a instrução requer a busca de operandos da memória. Dessa forma:
    - 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.
    - A memória lê a palavra de memória e transfere o conteúdo para o registrador MBR;
  - A UC sabe que a instrução ADD envolve a realização de uma operação de soma na ULA, então:
    - 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.
    - A ULA realiza a operação de soma.
    - 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.

# IAS

- Instruções e programação do IAS

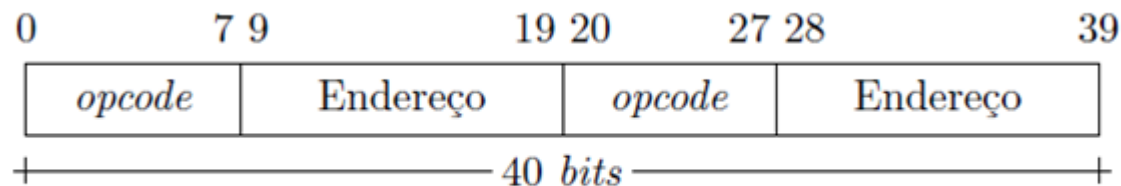


O conjunto de instruções do computador IAS possui 20 instruções. As instruções podem ser classificadas em 4 tipos distintos:

- **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.

# IAS

- Instruções de transferência de dados



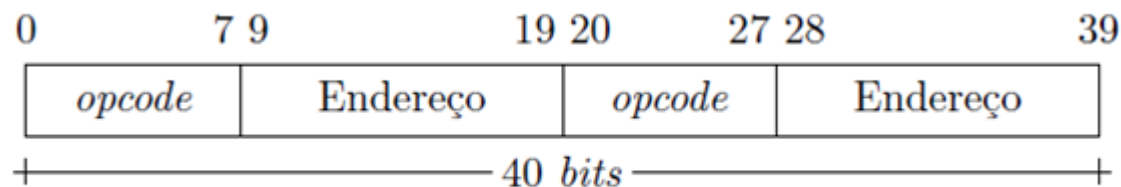
Duas instruções de 20 *bits* em uma palavra da memória

- Instrução LOAD M(X)

Sintaxe	Operação	Codificação		
LOAD M(X)	AC := Mem[X]	<table><tr><td>00000001</td><td>X</td></tr></table>	00000001	X
00000001	X			
Transfere o valor armazenado no endereço X da memória para o registrador AC.				

# IAS

- Instruções de transferência de dados



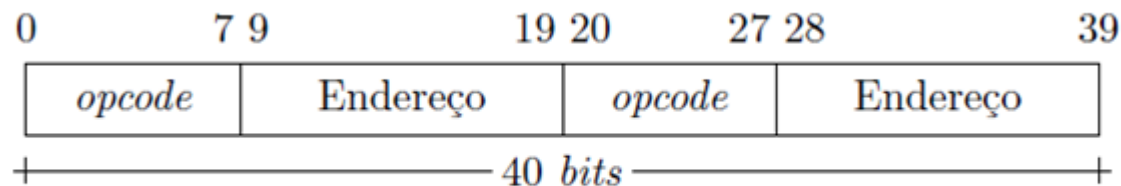
Duas instruções de 20 *bits* em uma palavra da memória

- Instrução LOAD MQ,M(X)

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

# IAS

- Instruções de transferência de dados



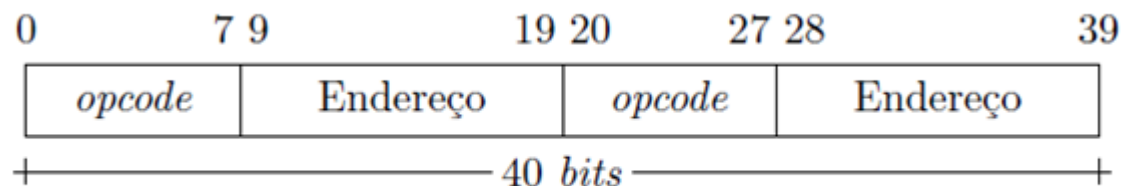
Duas instruções de 20 *bits* em uma palavra da memória

- Instrução STOR M(X)

Sintaxe	Operação	Codificação		
STOR M(X)	Mem[X] := AC	<table><tr><td>00100001</td><td>X</td></tr></table>	00100001	X
00100001	X			
Transfere o conteúdo do registrador AC para a palavra da memória no endereço X.				

# IAS

- Instruções de transferência de dados



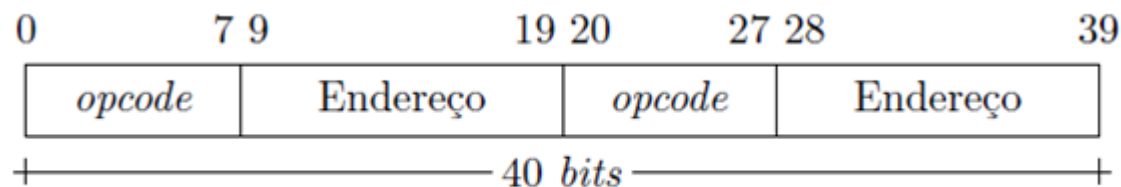
Duas instruções de 20 *bits* em uma palavra da memória

- Instrução LOAD MQ

Sintaxe	Operação	Codificação		
LOAD MQ	AC := MQ	<table><tr><td>00001010</td><td>—</td></tr></table>	00001010	—
00001010	—			
Transfere o conteúdo do registrador MQ para o registrador AC.				

# IAS

- Instruções de transferência de dados



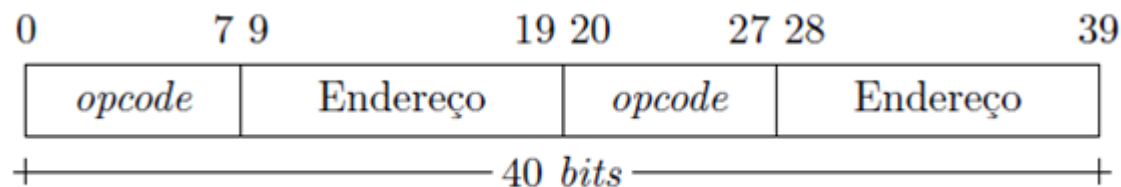
Duas instruções de 20 *bits* em uma palavra da memória

- Instrução LOAD |M(X)|

Sintaxe	Operação	Codificação		
LOAD  M(X)	AC :=  Mem[X]	<table><tr><td>00000011</td><td>X</td></tr></table>	00000011	X
00000011	X			
Transfere o absoluto do valor armazenado no endereço X da memória para o registrador AC.				

# IAS

- Instruções de transferência de dados



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

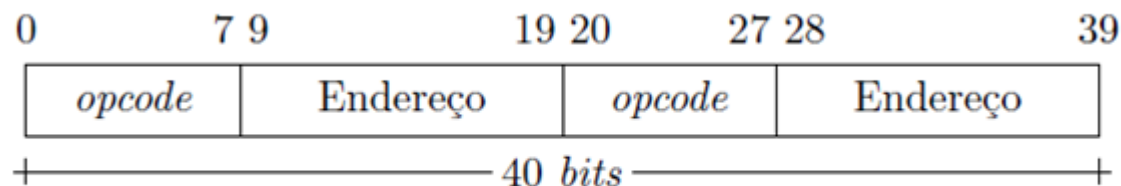
- Instrução LOAD -M(X)

Sintaxe	Operação	Codificação		
LOAD -M(X)	AC := -(Mem[X])	<table border="1"><tr><td>00000010</td><td>X</td></tr></table>	00000010	X
00000010	X			
Transfere o negativo do valor armazenado no endereço X da memória para o registrador AC.				



# IAS

- Instruções aritméticas



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

- Instrução ADD M(X)

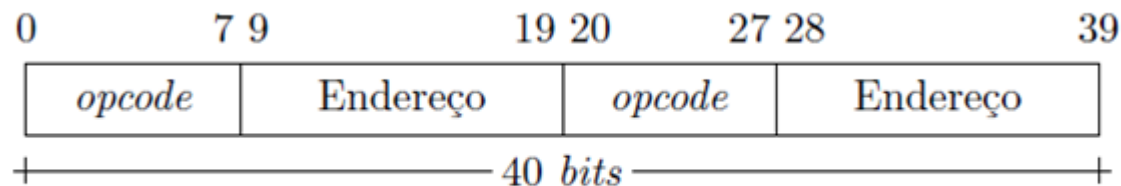
Sintaxe	Operação	Codificação		
ADD M(X)	AC := AC + Mem[X]	<table border="1"><tr><td>00000101</td><td>X</td></tr></table>	00000101	X
00000101	X			
Soma o valor armazenado no endereço X da memória com o valor armazenado no registrador AC e armazena o resultado no registrador AC.				

```

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

# IAS

- Instruções aritméticas



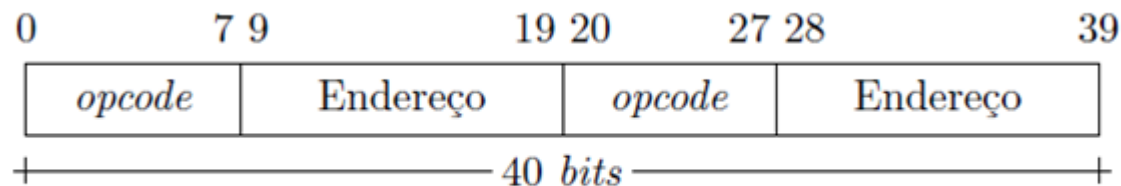
Duas instruções de 20 *bits* em uma palavra da memória

- 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.				

# IAS

- Instruções aritméticas



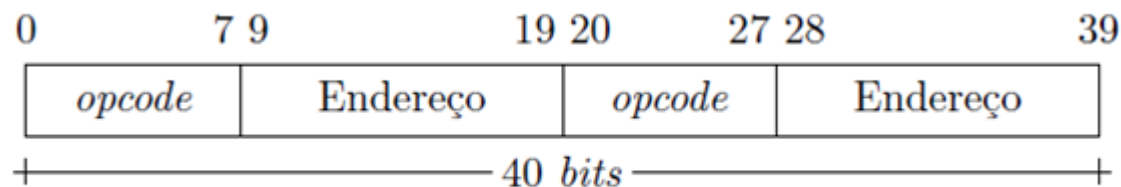
Duas instruções de 20 *bits* em uma palavra da memória

- Instrução SUB M(X)

Sintaxe	Operação	Codificação		
SUB M(X)	AC := AC - Mem[X]	<table border="1"><tr><td>00000110</td><td>X</td></tr></table>	00000110	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.				

## IAS

- Instruções aritméticas



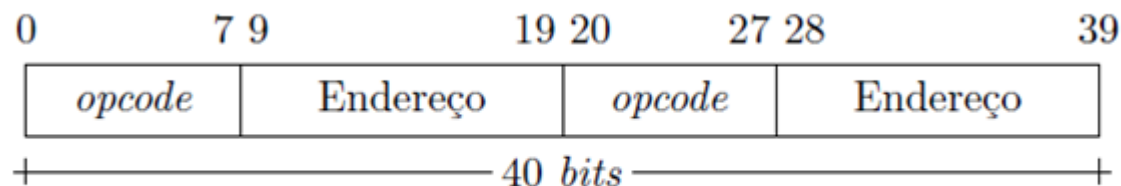
Duas instruções de 20 *bits* em uma palavra da memória

- Instrução SUB |M(X)|

Sintaxe	Operação	Codificação		
SUB  M(X)	AC := AC -  Mem[X]	<table border="1"><tr><td>00001000</td><td>X</td></tr></table>	00001000	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.				

## IAS

- Instruções aritméticas



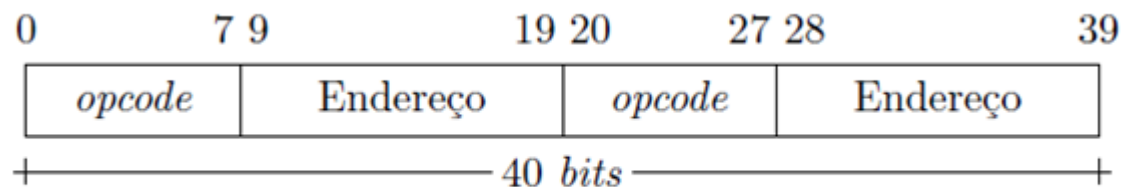
Duas instruções de 20 *bits* em uma palavra da memória

- Instrução MUL M(X)

Sintaxe	Operação	Codificação		
MUL M(X)	AC:MQ := MQ * Mem[X]	<table border="1"><tr><td>00001011</td><td>X</td></tr></table>	00001011	X
00001011	X			
<p>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 <i>bits</i>. Os 40 <i>bits</i> mais significativos são armazenados em AC e os 40 <i>bits</i> menos significativos são armazenados em MQ.</p>				

# IAS

- Instruções aritméticas



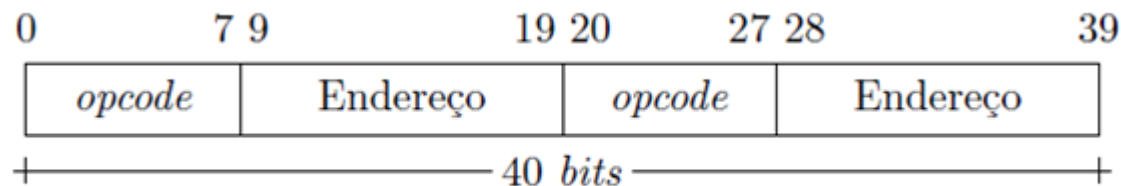
Duas instruções de 20 *bits* em uma palavra da memória

- Instrução DIV M(X)

Sintaxe	Operação	Codificação
DIV M(X)	$MQ := AC / Mem[X]$ $AC := AC \% Mem[X]$	<div>00001100</div> <div>X</div>
Divide o valor em AC pelo valor armazenado no endereço X da memória. Coloca o quociente em MQ e o resto em AC.		

# IAS

- Instruções aritméticas



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

- Instrução RSH

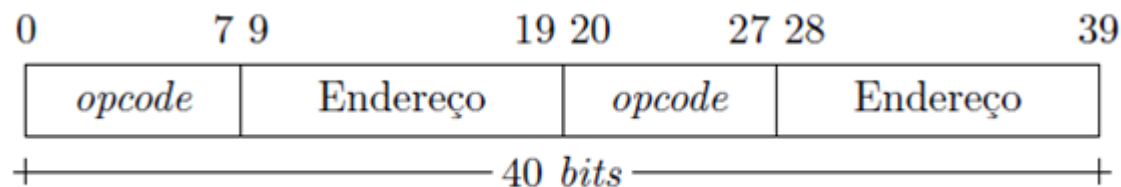
Sintaxe	Operação	Codificação		
RSH	$AC := AC \gg 1$	<table border="1"><tr><td>00010101</td><td>X</td></tr></table>	00010101	X
00010101	X			
Desloca os <i>bits</i> do registrador AC para a direita.				

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

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

# IAS

- Instruções aritméticas



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

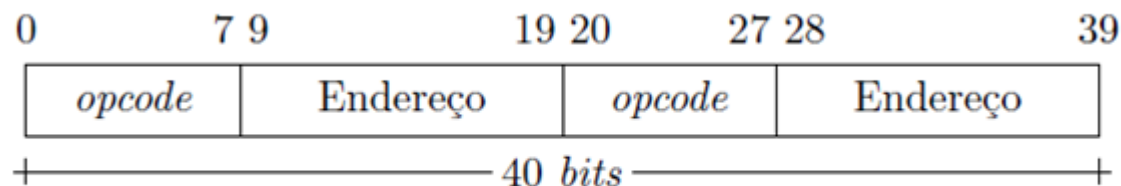
- Instrução LSH

Sintaxe	Operação	Codificação		
LSH	$AC := AC \ll 1$	<table border="1"><tr><td>00010100</td><td>X</td></tr></table>	00010100	X
00010100	X			
Desloca os <i>bits</i> do registrador AC para a esquerda.				



# 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.	<table border="1"><tr><td>00001101</td><td>X</td></tr></table>	00001101	X
00001101	X			
Salta para a instrução à esquerda ( <i>bits</i> 0 a 19) da palavra de memória armazenada no endereço M(X).				

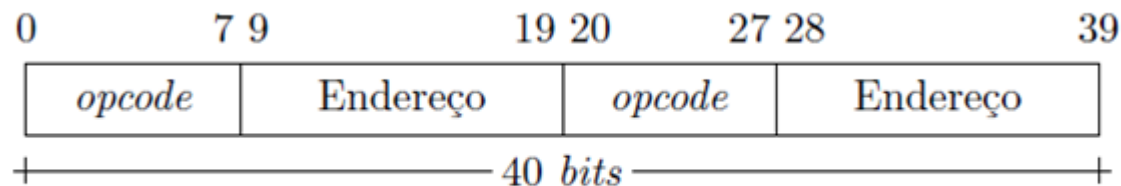
```

000  LOAD M(100);      ADD  M(101)
001  STOR M(100);     JUMP M(000,0:19)

```

## IAS

- Instruções de salto



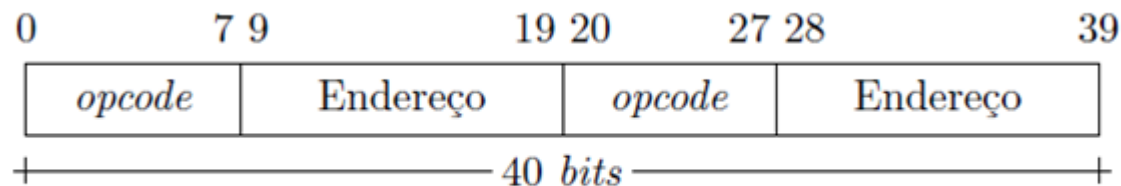
Duas instruções de 20 *bits* em uma palavra da memória

- Instrução JUMP M(X,20:39)

Sintaxe	Operação	Codificação		
JUMP M(X,20:39)	PC := M(X) e executa instrução à direita.	<table border="1"><tr><td>00001110</td><td>X</td></tr></table>	00001110	X
00001110	X			
Salta para a instrução à direita ( <i>bits</i> 20 a 39) da palavra de memória armazenada no endereço M(X).				

## 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.	<table border="1"><tr><td>00001111</td><td>X</td></tr></table>	00001111	X
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); ...

```

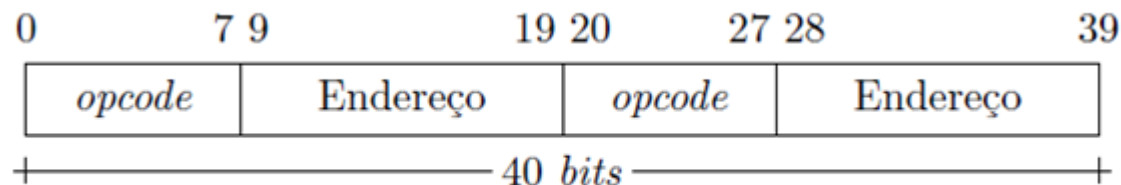
```

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

```

## IAS

- Instruções de salto



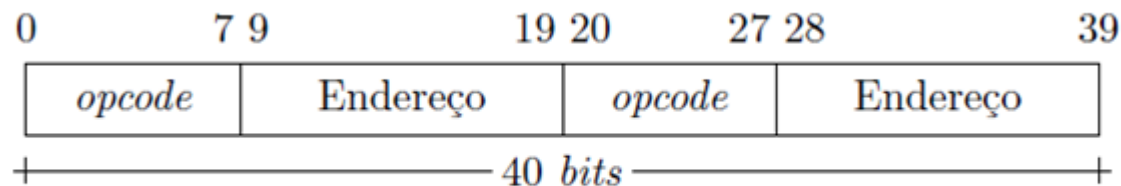
Duas instruções de 20 *bits* em uma palavra da memória

- 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.	<table border="1"><tr><td>00010000</td><td>X</td></tr></table>	00010000	X
00010000	X			
Salta para a instrução à direita ( <i>bits</i> 20 a 39) da palavra de memória se o valor armazenado em AC for maior ou igual à zero.				

## 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.	<table><tr><td>00010000</td><td>X</td></tr></table>	00010000	X
00010000	X			
Salta para a instrução à direita ( <i>bits</i> 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  ...

```

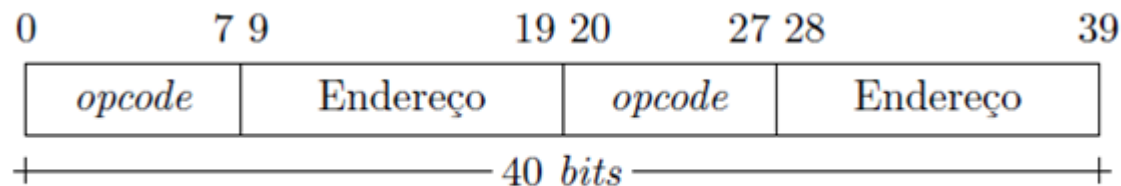
```

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

```

## IAS

- Instruções de modificação de endereço



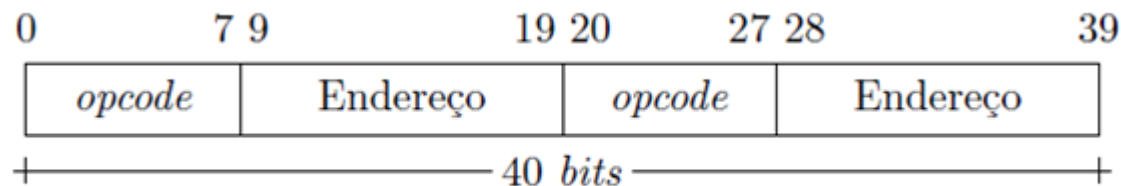
Duas instruções de 20 *bits* em uma palavra da memória

```
int A[1024];

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

## IAS

- Instruções de modificação de endereço



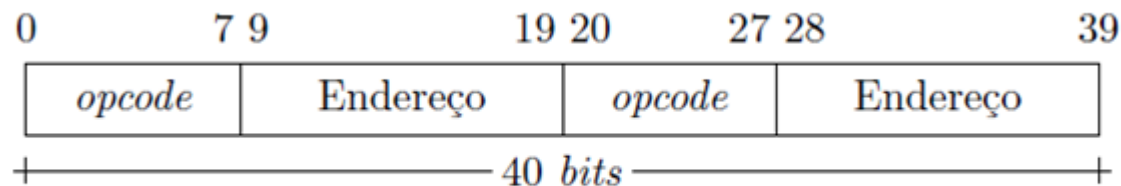
Duas instruções de 20 *bits* em uma palavra da memória

- Instrução STOR M(X,8:19)

Sintaxe	Operação	Codificação		
STOR M(X,8:19)	Mem[X](8:19) := AC(28:39)	<table><tr><td>00010010</td><td>X</td></tr></table>	00010010	X
00010010	X			
Move os 12 <i>bits</i> à direita do registrador AC para o campo endereço da instrução à esquerda da palavra de memória no endereço X.				

## 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)	<table border="1"><tr><td>00010011</td><td>X</td></tr></table>	00010011	X
00010011	X			
Move os 12 <i>bits</i> à direita do registrador AC para o campo endereço da instrução à direita da palavra de memória no endereço X.				

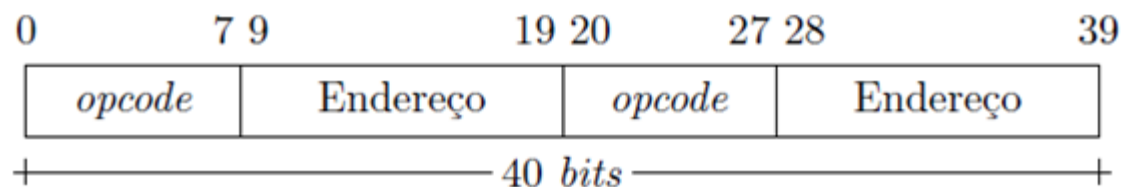


## IAS

- 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.

## IAS

- Resolução do exercício



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

1	Endereço	Instruções / Dados	
2			
3	000	LOAD M(0F2); STOR M(002,28:39)	# Modifica o endereço da instrução ADD
4	001	ADD M(0F1); STOR M(0F2)	# e atualiza o apontador.
5	002	LOAD M(0F3); ADD M(000)	# Carrega a variável e soma com o conteúdo
6			# do vetor apontado pelo apontador.
7	003	STOR M(0F3); LOAD M(0F0)	# Salva a soma e carrega o contador de it.
8	004	SUB M(0F1); STOR M(0F0)	# Atualiza o contador de iterações.
9	005	JUMP+ M(000,0:19); ...	
10			
11	0F0	00 00 00 00 19	# Contador de iterações
12	0F1	00 00 00 00 01	# Constante 1
13	0F2	00 00 00 01 00	# Apontador
14	0F3	00 00 00 00 00	# variável soma
15			
16	100	00 00 00 00 00	# Primeiro elemento do vetor
17	...	...	

## IAS


- Exercício proposto 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.

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

## IAS

- Linguagem de montagem do IAS

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

## 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					

## 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						
4		102	00	00	00	00	01
5	.org 0x000	103	00	00	00	00	0A
6	LOAD M(0x102)						
7	MUL M(0x103)						
8	LOAD MQ						
9	JUMP M(0x000,0:19)						
10							
11	Ling. de Montagem	Mapa de memória					

# 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)
```



# 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:
```

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

## 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 á 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	<code>.org 0x100</code>	100	00	00	00	01	01
2	<code>base:</code>	101	00	00	00	00	00
3	<code>.word vetor</code>	102	00	00	00	00	01
4	<code>vetor:</code>	103	00	00	00	00	02
5	<code>.word 00 00 00 00 00</code>						
6	<code>.word 00 00 00 00 01</code>						
7	<code>.word 00 00 00 00 02</code>						
8	<code>fim_vetor:</code>						
9							
10	Linguagem de Montagem	Mapa de memória					

## 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

```

---

# 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
```

## 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	<code>.set CODIGO 0x000</code>	000 0D 00 00 00 00
2	<code>.set DADOS 0x100</code>	100 00 00 00 00 05
3	<code>.set TAMANHO 200</code>	101 00 00 00 00 05
4		...
5	<code>.org CODIGO</code>	1C6 00 00 00 00 05
6	<code>laco:</code>	1C7 00 00 00 00 05
7	<code>JUMP M(laco)</code>	
8		
9	<code>.org DADOS</code>	
10	<code>vetor:</code>	
11	<code>.wfill TAMANHO, 0x5</code>	

13      Ling. de Montagem

Mapa de memória

1	<code>.org 0x000</code>	000 0D 00 00 00 00
2	<code>laco:</code>	100 00 00 00 00 05
3	<code>JUMP M(laco)</code>	101 00 00 00 00 05
4		...
5	<code>.org 0x100</code>	1C6 00 00 00 00 05
6	<code>vetor:</code>	1C7 00 00 00 00 05
7	<code>.wfill 200, 0x5</code>	

9      Ling. de Montagem

Mapa de memória

## IAS

- Exercício proposto 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.