ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Linguagem de programação

- **Linguagem de Alto Nível – próximo ao ser humano, escrita de forma textual.**
  - **Ex: a=a+;**
- **Linguagem de Montagem (Assembly) – próximo à linguagem de máquina, escrita em códigos (mnemônicos)**
  - **Ex: ADD AX,BX;**
- **Linguagem de Máquina – linguagem que o computador consegue executar – códigos binários**
  - **Ex: 01010001**

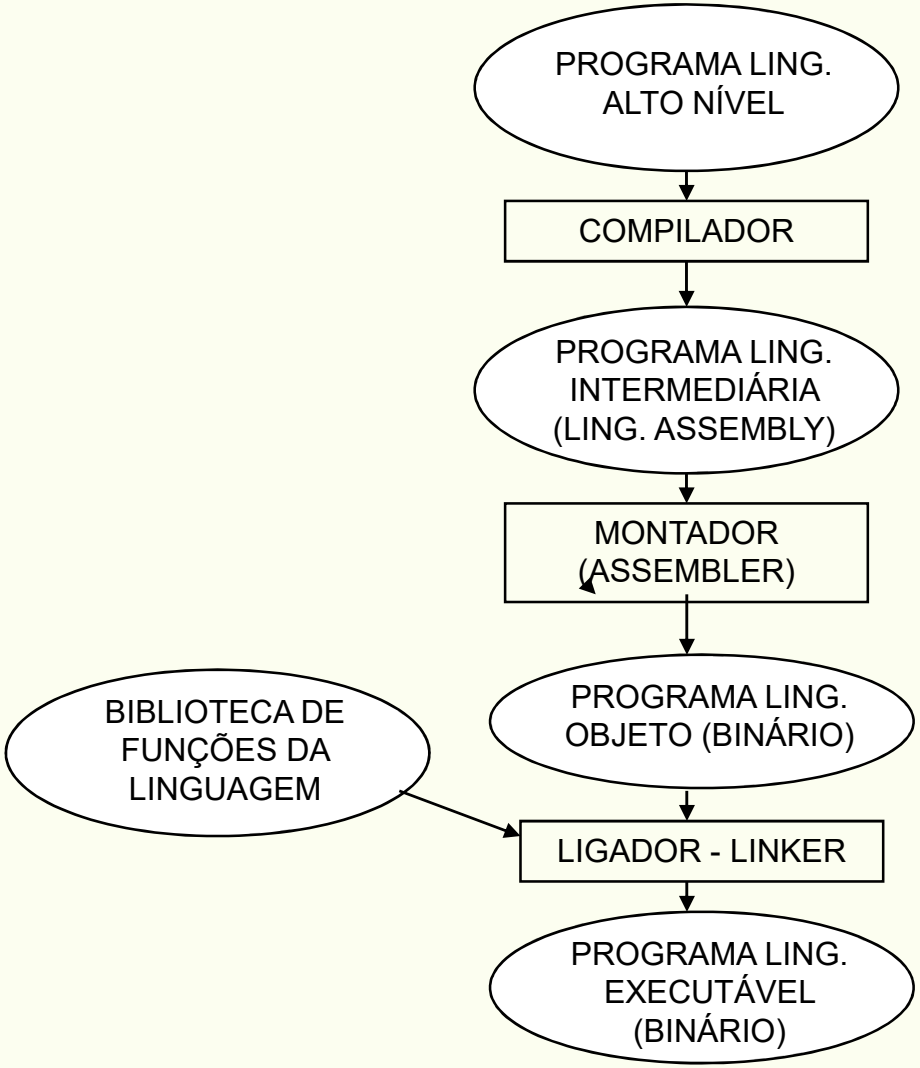
ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Execução de um programa

- **Um programa escrito em linguagem de alto nível, para ser executado ele deve:**
  - **Ser traduzido para linguagem de máquina (compiladores);**
  - **Ter seus endereços realocados, conforme posição onde será carregado na memória (loaders);**
  - **Alocá-lo em um região da memória (loaders).**

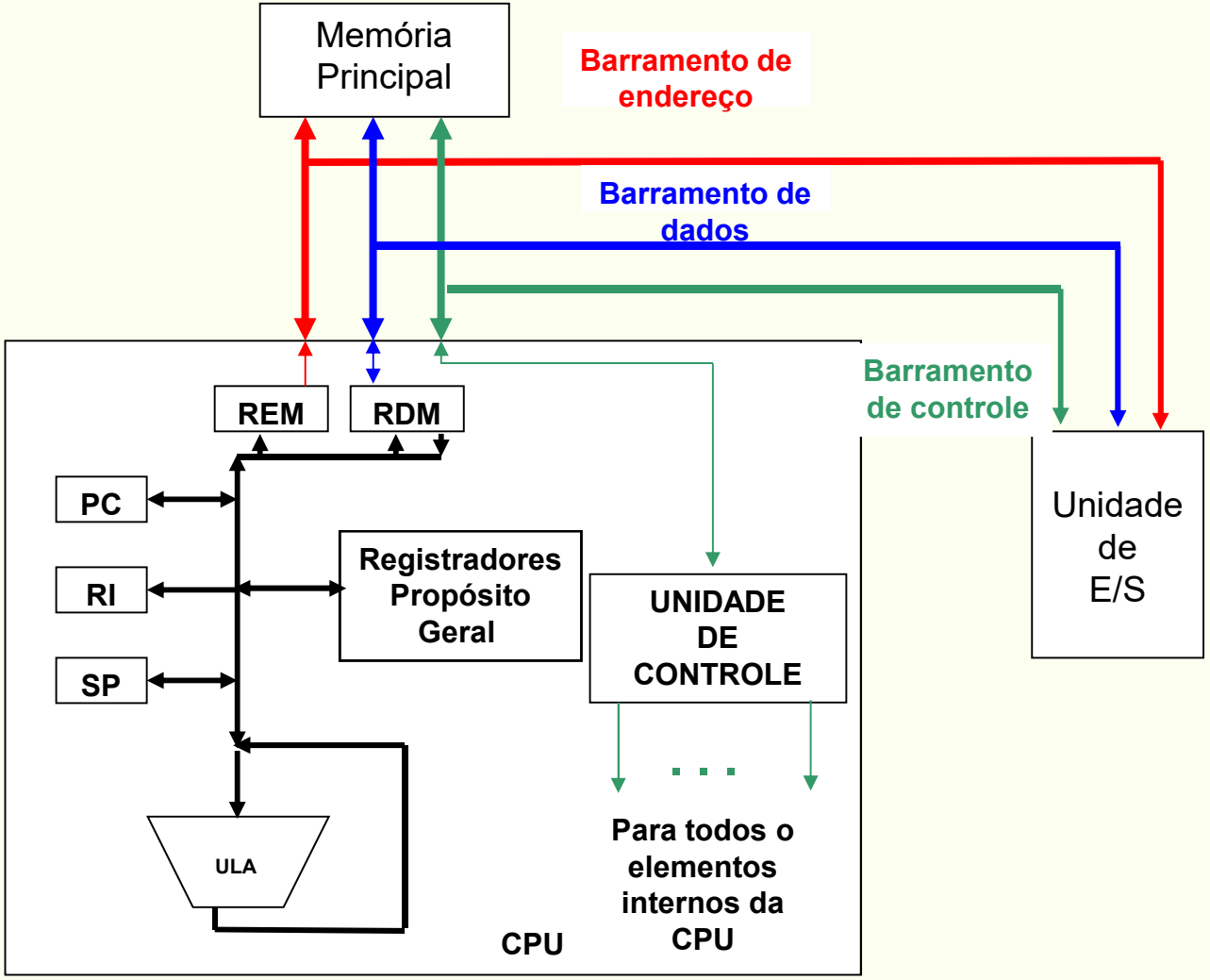
ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Processo de tradução de um programa em linguagem de alto nível



ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Organização Básica de um Computador Digital



ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Organização Básica de um Computador Digital

- **Unidade Central de Processamento – CPU:**
  - **Unidade de Controle – UC;**
  - **Unidade Lógica e Aritmética – ULA;**
  - **Registradores de Propósito Geral – GPR;**
  - **Registradores Específicos.**

- **Unidade de Memória → hierarquia de memória:**
  - **Memória Principal;**
  - **Memória Secundária;**

- **Unidade de Entrada e Saída:**
  - **Interfaces;**
  - **Canais de E/S;**
  - **Processadores E/S.**

- **Barramentos:**
  - **Barramento de Endereços;**
  - **Barramento de Dados;**
  - **Barramento de Controle.**

Organização Básica de um Computador Digital

- **Unidade Central de Processamento – CPU**
  - **Responsável por todo o processamento (execução de programas) no sistema**
    - **Unidade de Controle: circuito que gera os sinais de controle responsáveis pelo gerenciamento (controle) de todas as atividades do computador.**
    - **Unidade Lógica e Aritmética – ULA: circuito responsável por efetuar todas as operações lógicas e aritméticas.**
    - **Registradores de Propósito Geral – GPR: elementos de memória (circuitos) responsáveis por armazenar os dados que são utilizados durante a execução de um programa (instruções).**

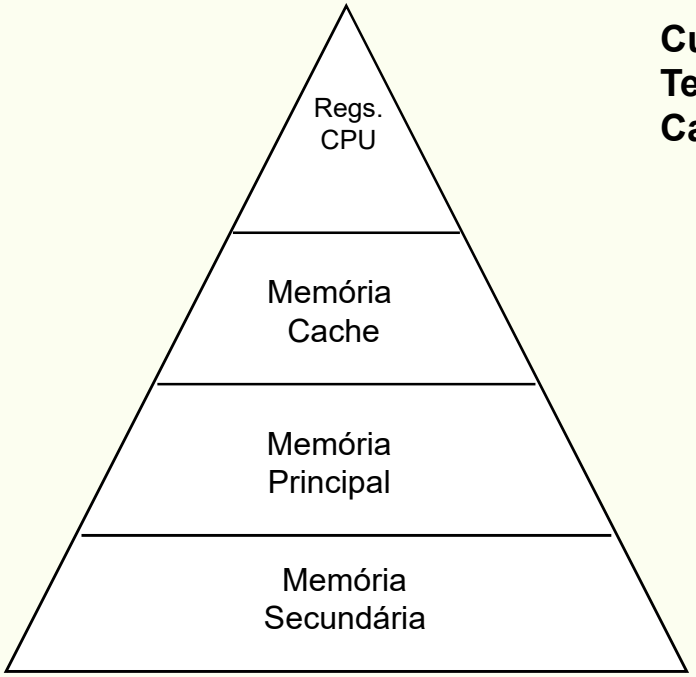
Organização Básica de um Computador Digital

- **Unidade Central de Processamento – CPU (cont.)**
  - **Registradores Específicos:**
    - **Program Counter – PC: armazena o endereço da próxima instrução a ser executada;**
    - **Stack Pointer – SP: armazena o endereço do topo da pilha;**
    - **Registrador de Instrução – RI: armazena a instrução que está sendo executada;**
    - **Registrador de Dados de Memória – RDM: armazena os dados que vem da memória (lidos) ou que vão para a memória (escritos);**
    - **Registrador de Endereços de memória – REM: armazena o endereço enviado para a memória, quando ocorrer um acesso à mesma (leitura ou escrita)**

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Organização Básica de um Computador Digital

- **Unidade de Memória**
  - **Hierarquia de Memória: sistema de memória com objetivo de melhorar o desempenho de um sistema computacional, diminuindo o tempo de acesso médio**



Custo de armazenamento por bit maior
Tempo de acesso menor
Capacidade de armazenamento menor

Custo de armazenamento por bit menor
Tempo de acesso menor
Capacidade de armazenamento maior

Organização Básica de um Computador Digital

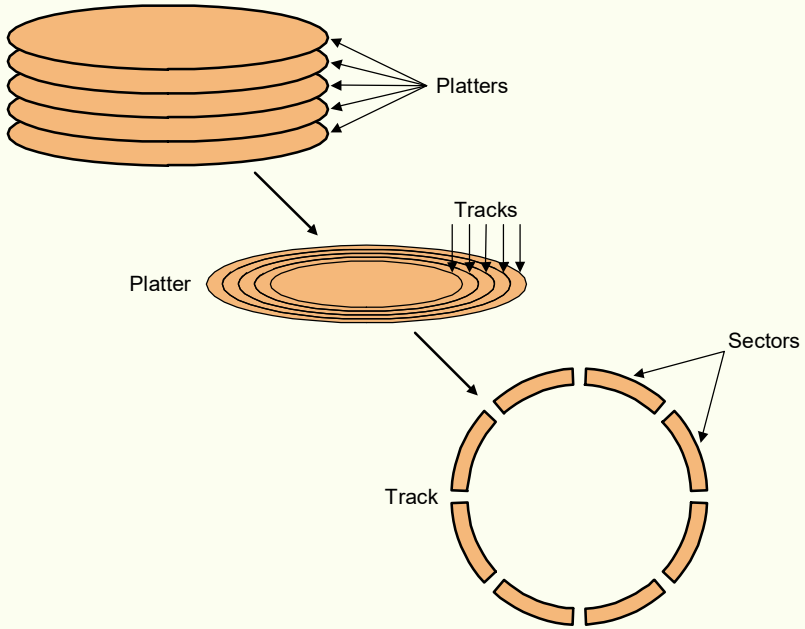
- **Memórias**
  - **Semicondutoras: fabricadas com materiais semicondutores (silício) – circuitos integrados.**
    - **RAM – Random Access Memory : memória de acesso aleatório, volátil.**
      - **SRAM – RAM estática: seu conteúdo só se altera quando se escreve nela ou quando se desliga a tensão de alimentação. Exemplo – registradores da CPU, memória cache.**
      - **DRAM – RAM dinâmica: periodicamente é necessário reescrever o seu conteúdo (refresh de memória) pois há diminuição de cargas elétricas.**

        Exemplo – memória principal.
    - **ROM – Read Only Memory: memória somente de leitura, não volátil.**
      - **ROM: gravação feita pelo fabricante da memória, não apagável;**
      - **PROM – Programmable ROM: programação feita pelo usuário, não apagável;**
      - **EPROM – Erasable PROM: programação feita pelo usuário, apagável através de luz ultra-violeta;**
      - **EEPROM – Electrical EPROM: programação feita pelo usuário, apagável eletricamente;**
    - **Flash – memória semicondutora, não volátil e de escrita e leitura, apagável.**

Organização Básica de um Computador Digital

- **Memórias (continuação)**
  - **Magnéticas**
    - **Discos – Hard Disk – HDs**
    - **Opticos – CD-ROM, DVD, etc.**
    - **Fitas – cartchos, rolos, etc.**

Exemplo: memórias secundárias



Disco Magnético ➔ pratos, lados, trilhas e setores

Organização Básica de um Computador Digital

- **Unidade de Entrada e Saída: responsável por gerenciar a ligação entre CPU-Memória-barramentos e os periféricos.**
  - **Interfaces – circuitos simples que apenas compatibilizam a comunicação (protocolo). O controle da transferência é feita pela CPU. Exemplo: interface serial RS232, interface paralela, interface USB;**
  - **Canais de E/S – circuitos que controlam e compatibilizam a comunicação. A CPU apenas inicia a transferência. Exemplo – Controlador de Acesso Direto à Memória (DMA – Direct Access Memory);**
  - **Processadores de E/S – são CPUs dedicadas a fazer E/S de dados. Iniciam e controlam a comunicação.**

Organização Básica de um Computador Digital

- **Barramentos: Conjunto de fios que faz a ligação física entre as diversas unidades.**
  - **Barramento de Endereços: Por onde trafegam os endereços;**
  - **Barramento de Dados: Por onde trafegam os dados;**
  - **Barramento de Controle: por onde trafegam os sinais de controle;**

- **Observação:**

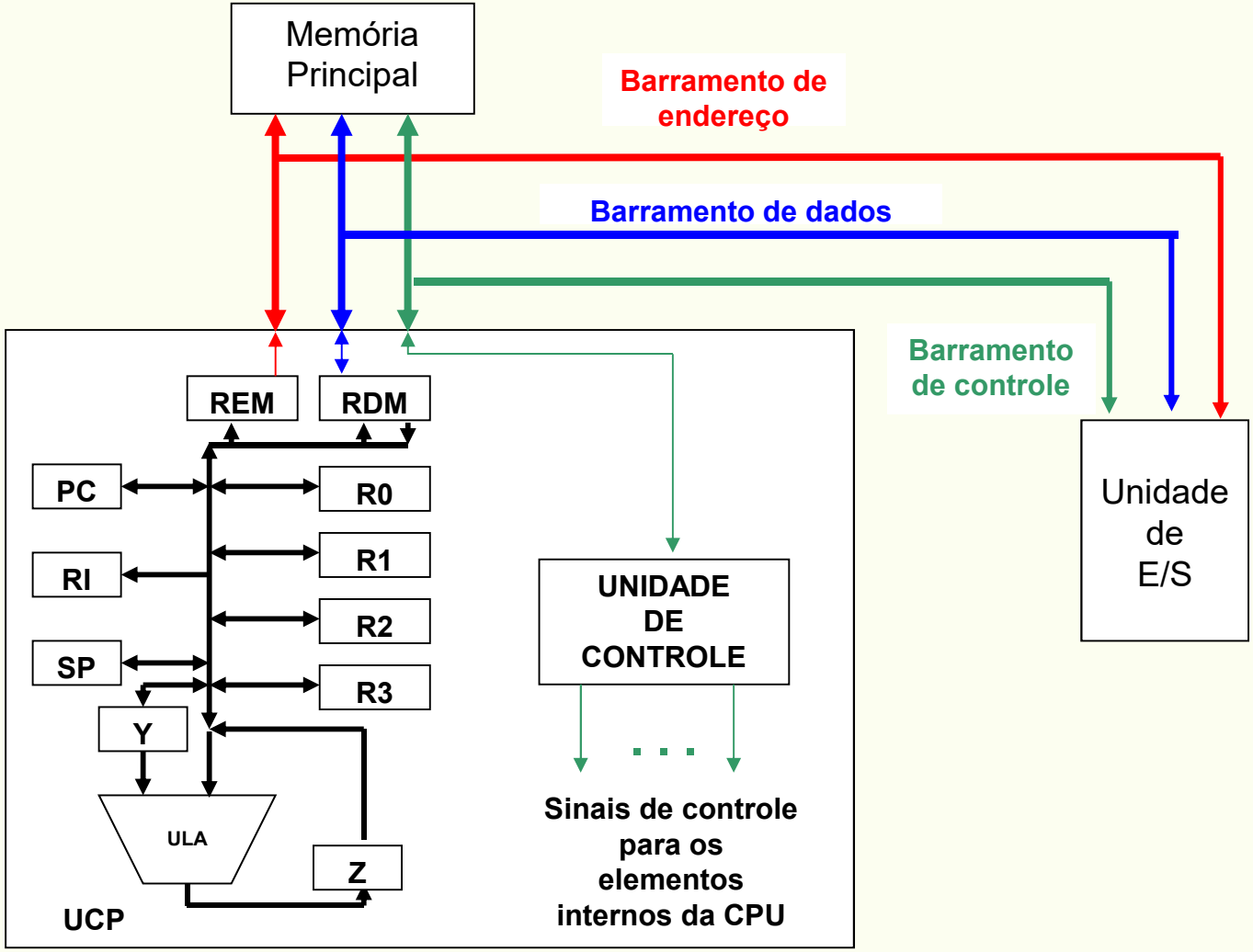
  Internamente à CPU, existe um barramento interno de dados que liga os registradores com a ULA e a UC, e um barramento interno de controle que liga a UC a todos os elementos da CPU.

Arquitetura de um Computador Digital

- **Formato das Instruções**
  - **Tamanho (número de bits) e o significado de cada campo de bits de uma instrução de linguagem de máquina.**
- **Conjunto de Instruções**
  - **Cada processador tem o seu conjunto de instruções de linguagem de máquina (ISA – Instruction Set Architecture). Este conjunto contém todas as instruções, em linguagem de máquina, que o processador pode executar.**

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

ESTUDO DE CASO - CPU HIPOTÉTICA 1



ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Execução de uma instrução pela CPU

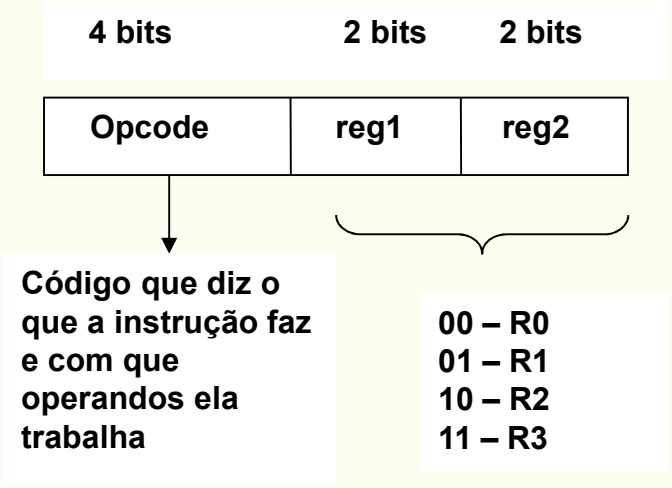
- **Ciclo de execução de uma instrução:**
  - **Leitura da instrução da memória principal – Fetch da Instrução**
    REM ← PC
    Read (sinal de controle)
    PC ← PC atualizado
    RDM ← MEM[REM] (instrução lida)
  - **Decodificação da instrução**
    RI ← RDM (instrução)
    É feita a decodificação pela Unidade de Controle
  - **Busca dos operandos da instrução na memória – se houver**
    REM ← PC
    Read (sinal de controle)
    PC ← PC atualizado
    RDM ← MEM[REM] (operando lido)
  - **Execução da instrução – depende da instrução**
- **Obs – Quando usamos [..], significa que estamos acessando um conteúdo de memória, cujo endereço está dentro dos colchetes. MEM[...]**

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

ESTUDO DE CASO - CPU HIPOTÉTICA

- **Formatos das instruções da CPU HIPOTÉTICA:**

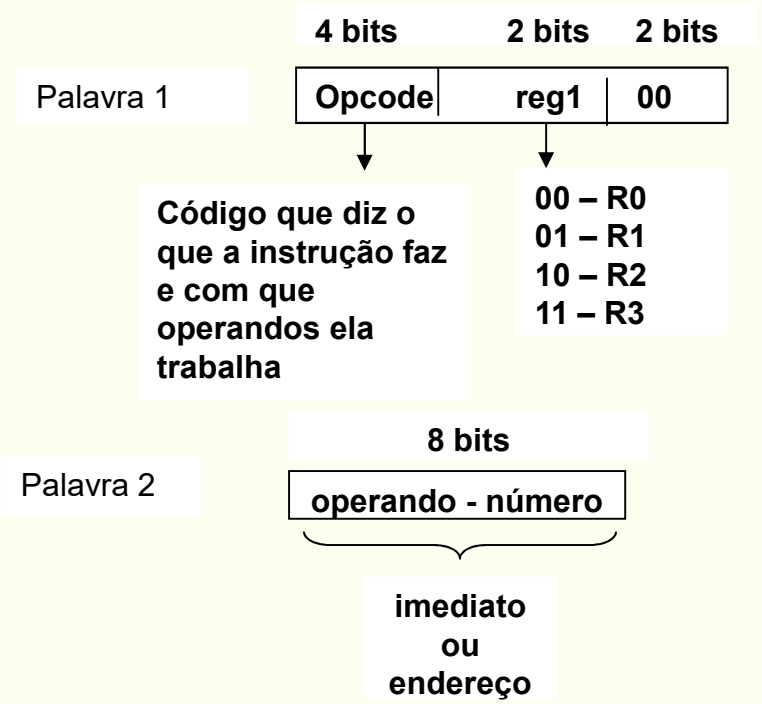
  – **Formato tipo I – Uma palavra de 8 bits, com os seguintes campos:**



Exemplo: MOV R0,R1 ; R0 ← R1

ESTUDO DE CASO - CPU HIPOTÉTICA

- **Formatos das instruções da CPU HIPOTÉTICA:**
  - **Formato tipo II – Duas palavras de 8 bits, com os seguintes campos:**



Exemplos:

MOV R0, 5 ; R0 ← 5

MOV R0, [5] ; R0 ← MEM[5]

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Mnemônico	Operandos	Opcode	Significado
Instruções de Movimentação de Dados			
MOV	Reg1,Reg2	0000	Reg1 ←Reg2
MOV	Reg,imed	1000	Reg ← imed
MOV	Reg,[end]	1001	Reg ← MEM[end]
MOV	[end],Reg	1010	MEM[end] ← Reg
Instruções Aritméticas e Lógicas			
ADD	Reg1,Reg2	0001	Reg1 ← Reg1 + Reg2
ADD	Reg,imed	1011	Reg ← Reg + imed
SUB	Reg1,Reg2	0010	Reg1 ← Reg1 - Reg2
SUB	Reg,imed	1100	Reg ← Reg – imed
AND	Reg1,Reg2	0011	Reg1 ← Reg1 *e* Reg2
AND	Reg,imed	1101	Reg ← Reg *e* imed
OR	Reg1,Reg2	0100	Reg1 ← Reg1 *ou* Reg2
Instruções de Manipulação de Pilha			
PUSH	Reg	0101	SP-- , MEM[SP] ← Reg
POP	Reg	0110	Reg←MEM[SP], SP++
Instruções de Controle de Fluxo de Execução			
JMP	end	1110	PC ← end
CALL	end	1111	SP-- , MEM[SP]←PC , PC←end
RET	---	0111	PC←MEM[SP] , SP++

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

MEMÓRIA

END	CONTEÚDO	
2	INSTR1	
3	INSTR2	
4	JMP 8	
5	INSTR4	
6	INSTR5	
7	INSTR6	
8	INSTR7	←PC
9	INSTR8	
10	INSTR9	

PROGRAMA COM 9 INSTRUÇÕES
INSTRX – 8 BITS

MEMÓRIA

END	CONTEÚDO	
92		
93		
94		
95		
96		
97		
98	C	←SP
99	B	
100	A	

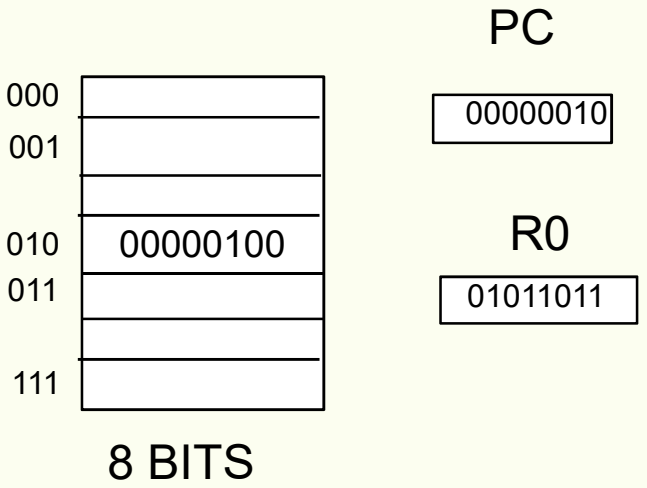
PILHA

Exercícios

- **Mostrar o ciclo de execução de instruções para todas as instruções do ISA da CPU Hipotética 1.**
    1) **MOV R1,R0**
    2) **MOV R2,16**
    3) **MOV R3,[4]**
    4) **MOV [4],R2**
    5) **ADD R1,R2**
    6) **AND R0,10**
    7) **JMP 6**
    8) **PUSH R2**
    9) **POP R3**
    10) **CALL**
    11) **RET**

Exercícios

1 – MOV R1,R0 ; R1 ← R0



A) Fetch da Instrução

REM ← PC ;(010)

Read

PC ← PC +1 ;(011)

RDM ← MEM[REM]; RDM ← 0000 0100

B) Decodificação

RI ← RDM ; RI ← 00000100

Decodificação

C) Busca de Operandos

Não tem

D) Execução

R1 ← R0 ; R1 ← 010111011

Exercícios

2 – MOV R2,16 ; R2 ← 16

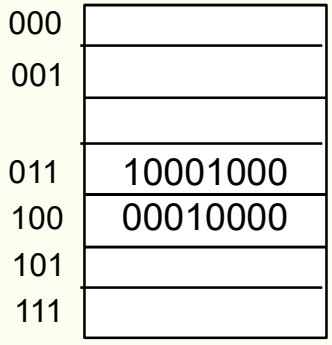
A) Fetch da Instrução

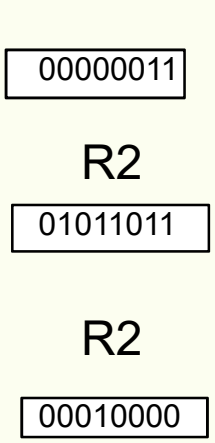
REM ← PC ; (011)

Read

PC ← PC +1 ; (100)

RDM ← MEM[REM]; RDM ← 10001000





B) Decodificação

RI ← RDM ; RI ← 10001000

Decodificação

C) Busca de Operandos

REM ← PC ; (100)

Read

PC ← PC +1 ; (101)

RDM ← MEM[REM] ; RDM ← 00010000

D)Execução

R2 ← RDM ; R2 ← 16

Exercícios

3 – MOV R3,[4] ; R3 ← MEM[4]

A) Fetch da Instrução

REM ← PC ;(000)
Read
PC ← PC +1 ;(001)
RDM ← MEM[REM] ;RDM ← 10011100

B) Decodificação

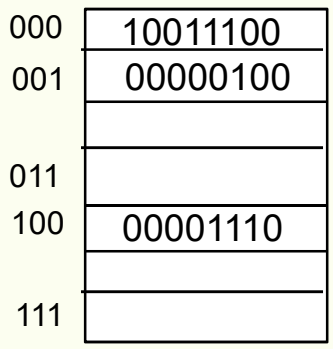
RI ← RDM ; RI ←10011100
Decodificação

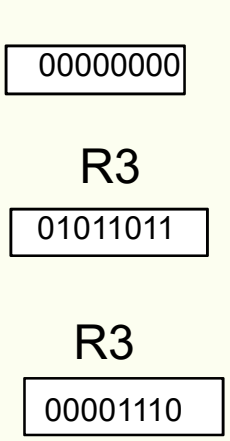
C) Busca de Operandos

REM ← PC ;(001)
Read
PC ← PC +1 ;(010)
RDM ← MEM[REM] ;RDM ← 00000100

D)Execução

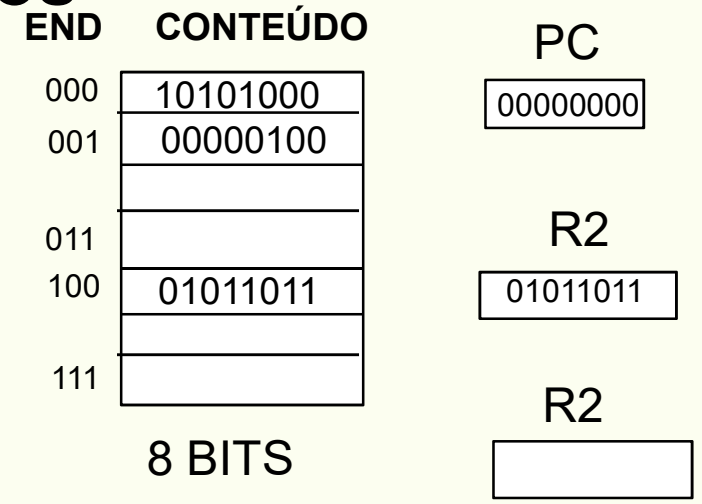
REM ← RDM ;(00000100)
Read
RDM ← MEM[REM] ;RDM ← 00001110
R3 ← RDM ;R3 ← 00001110





Exercícios

4 – MOV [4],R2 ; MEM[4] ← R2



Exercícios

4 – MOV [4],R2 ; MEM[4] ← R2

A) Fetch da Instrução

REM ← PC ; REM ← 000

Read

PC ← PC +1; PC ← 001

RDM ← MEM[REM] ;RDM ← 10101000

B) Decodificação

RI ← RDM ; RI ←10101000

decoficação

C) Busca de Operandos

REM ← PC ; REM ← 001

Read

PC ← PC + 1; PC ← 010

RDM ← MEM[REM] ; RDM ← 00000100

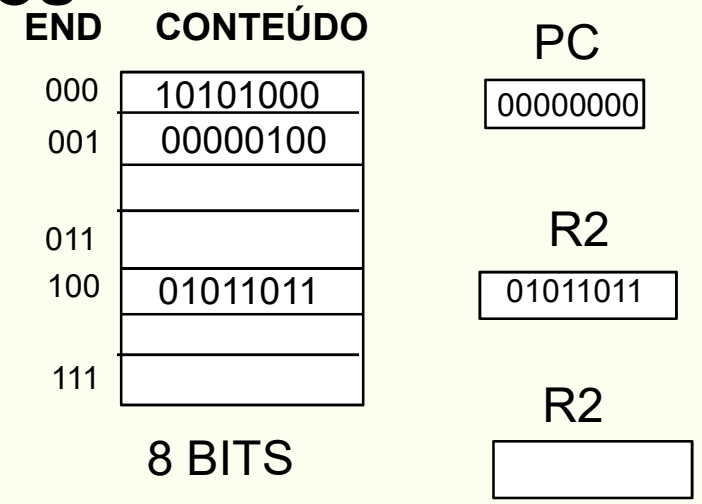
D)Execução

REM ← RDM ; REM ← 00000100

RDM ← R2 ; RDM ← 01011011

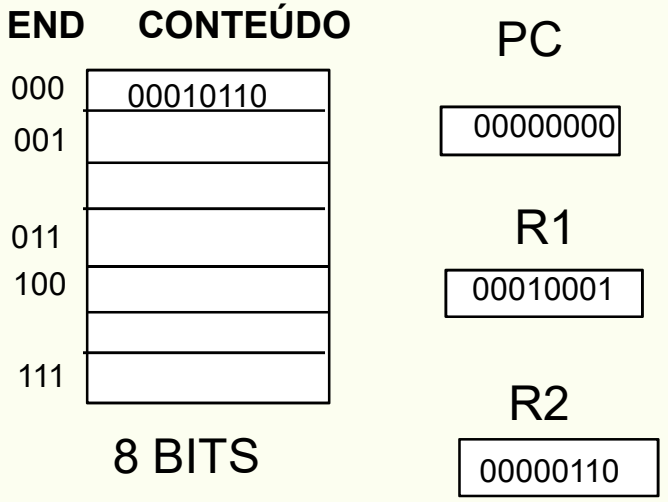
Write

MEM[REM] ← RDM ; MEM[4] ← 01011011



Exercícios

5 – ADD R1,R2 ; R1 ← R1 + R2



Exercícios

5 – ADD R1,R2 ; R1 ← R1 + R2

A) Fetch da Instrução

REM ← PC ; REM ← 000

Read

PC ← PC +1 ; PC ← 001

RDM ← MEM[REM] ; RDM ← 00010110

B) Decodificação

RI ← RDM ; RI <-- 00010110

Decodificação

C) Busca de Operandos

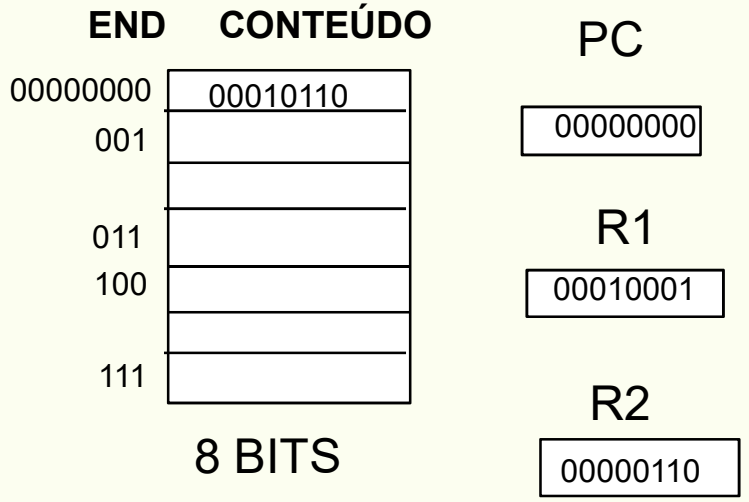
NÃO EXISTE

D)Execução

Y ← R1 ;Y ←00001011

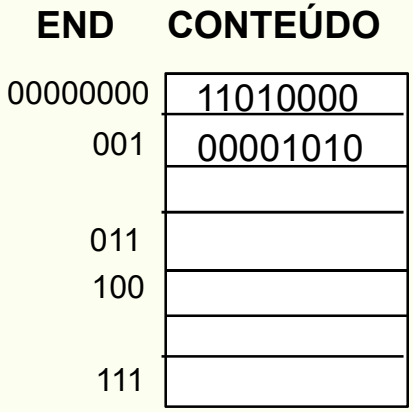
Z ← Y + R2 ;Z← 00001011 + 00000110

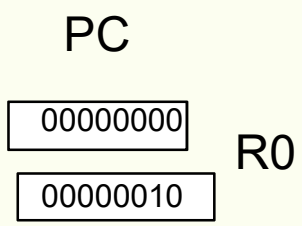
R1 ← Z ;R1 ← 00010001



ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

6- AND R0, 10 ; R0 ← R0 AND 00001010





OBS:

c = a OR b		
a	b	c
0	0	0
0	1	1
1	0	1
1	1	1
c = a AND b		
a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

6- AND R0, 10 ; R0 ← R0 AND 00001010

A) Fetch da Instrução

REM ← PC ; REM ← 000

Read

PC ← PC +1 ; PC ← 001

RDM ← MEM[REM] ; RDM ← 11010000

B) Decodificação

RI ← RDM ; RI ← 11010000

Decodificação

C) Busca de Operandos

REM ← PC ; REM ← 001

Read

PC ← PC +1 ; PC ← 010

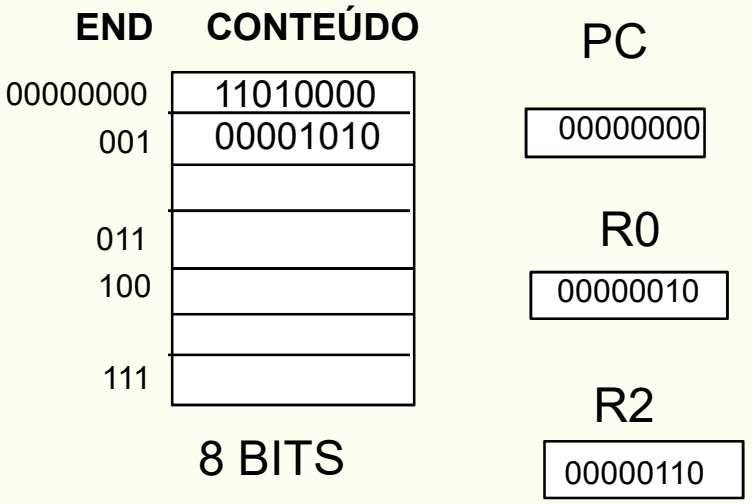
RDM ← MEM[REM] ; RDM ← 00001010

D)Execução

Y ← R0 ; Y← 00000010

Z← Y AND RDM ; Z ← 00000010 AND 00001010

R0 ← Z ; R0 ← 00000010

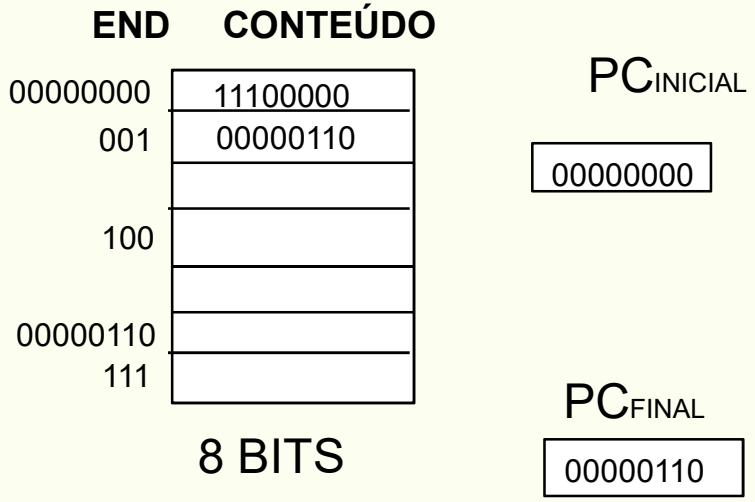


c = a OR b		
a	b	c
0	0	0
0	1	1
1	0	1
1	1	1

c = a AND b		
a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

Exercícios

7 – JMP 6 ; PC ← 00000110



ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

7 – JMP 6 ; PC ← 00000110

A) Fetch da Instrução

REM ← PC ; REM ← 000

Read

PC ← PC +1 ; PC ← 001

RDM ← MEM[REM] ; RDM ← 11100000

B) Decodificação

RI ← RDM ; RI ← 11100000

Decodificação

C) Busca de Operandos

REM ← PC ; REM ← 001

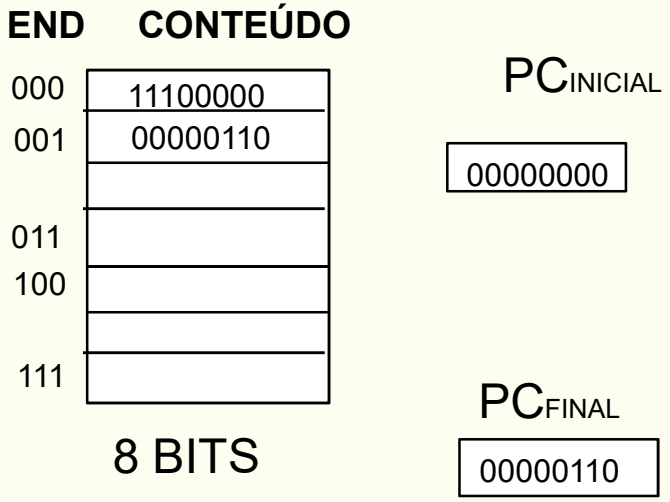
Read

PC ← PC +1 ; PC ← 010

RDM ← MEM[REM] ; RDM ← 00000110

D)Execução

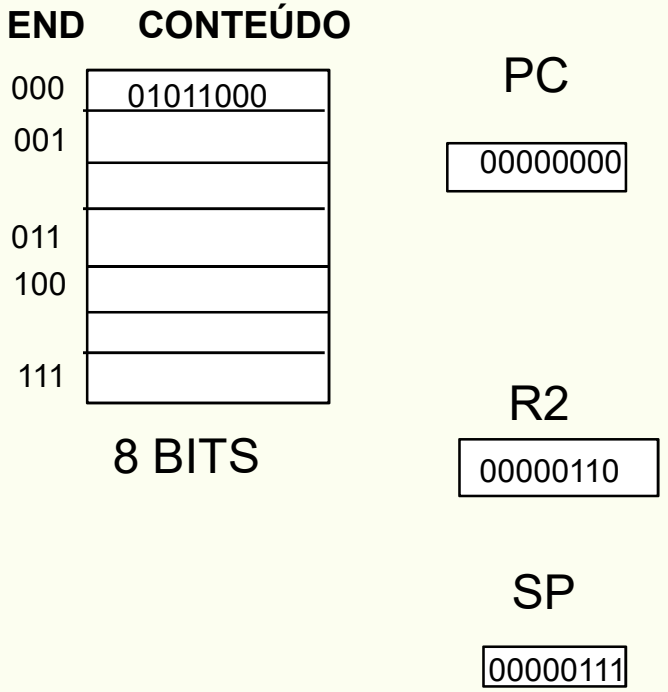
PC ← RDM ; PC ← 00000110



Exercícios

8 – PUSH R2 ; SP-- , [SP] <- R2

A)



Exercícios

8 – PUSH R2 ; SP-- , [SP] <- R2

A) Fetch da Instrução

REM <- PC ; REM <- 000

Read

PC <- PC +1 ; PC <- 001

RDM <- [REM] ; RDM <- 01011000

B) Decodificação

RI <- RDM ; RI <-

Decodificação

C) Busca de Operandos

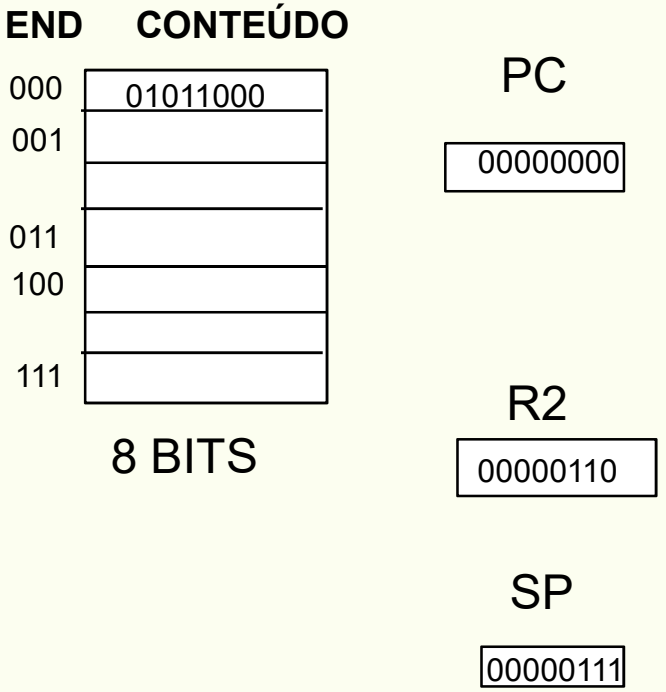
REM <- PC ; REM <- 001

Read

PC <- PC +1 ; PC <- 010

RDM <- [REM] ; RDM <- 00000110

D)Execução



Exercícios

8 – PUSH R2 ; SP-- , [SP] <- R2

A) Fetch da Instrução

REM <- PC ; REM <- 000

Read

PC <- PC +1 ; PC <- 001

RDM <- [REM] ; RDM <- 01011000

B) Decodificação

RI <- RDM ; RI <- 01011000

Decodificação

C) Busca de Operandos

NÃO EXISTE

D)Execução

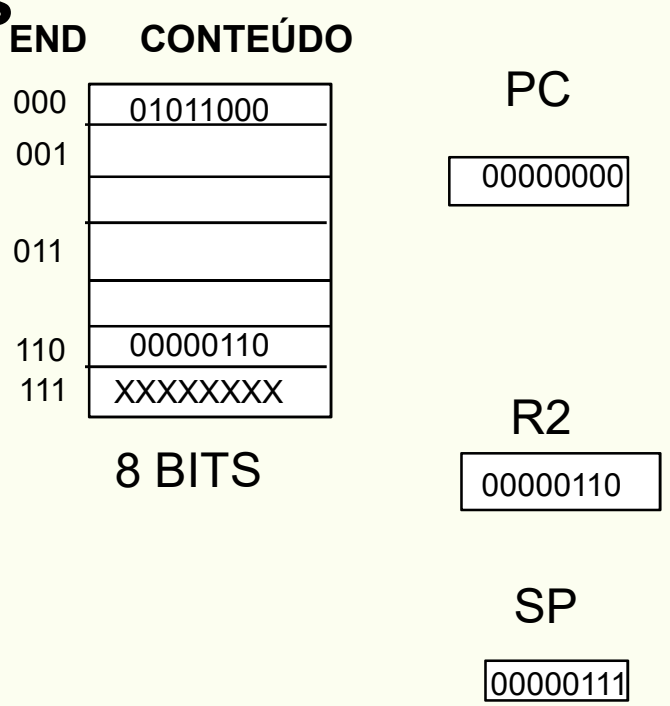
SP <- SP -1 ; SP <- 110

REM <- SP ; REM <- 110

RDM <- R2 ; RDM <- 00000110

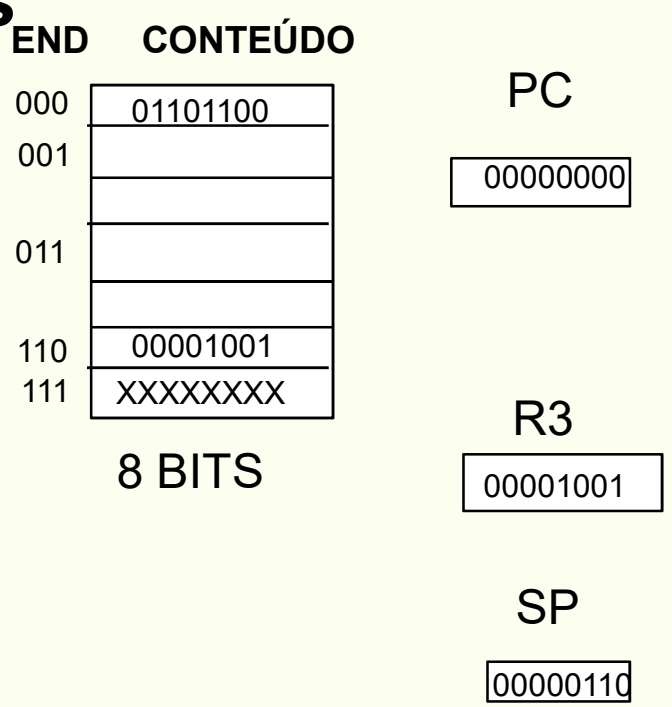
Write

[REM] <- RDM ; [110] <- 00000110



Exercícios

9 – POP R3 ; R3 <- [SP]; SP ++



Exercícios

9 – POP R3 ; R3 <- [SP]; SP ++

A) Fetch da Instrução

REM <- PC ; REM <- 000

Read

PC <- PC +1 ; PC <- 001

RDM <- [REM] ; RDM <- 01101100

B) Decodificação

RI <- RDM ; RI <- 01101100

Decodificação

C) Busca de Operandos

NÃO EXISTE

D)Execução

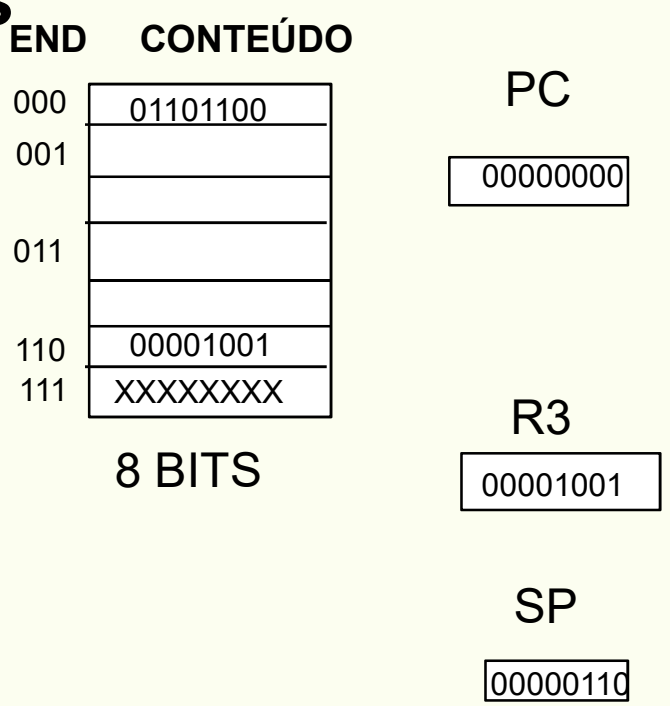
REM <- SP ; REM <- 110

Read

SP <- SP + 1 ; SP <- 111

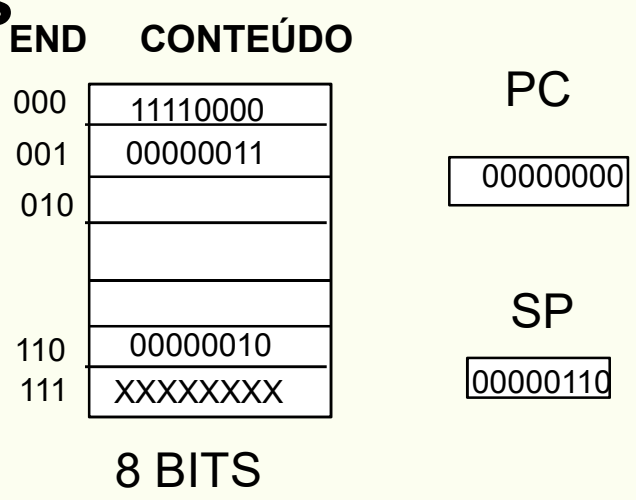
RDM <- [REM] ; RDM <- [110] ; RDM <- 00001001

R3 <- RDM ; R3 <- 00001001



Exercícios

10 – CALL 3; SP-- , [SP] <- PC, PC <- 3



Exercícios

10 – CALL 3; SP-- , [SP] <- PC, PC <- 3

A) Fetch da Instrução

REM <- PC ; REM <- 000

Read

PC <- PC +1 ; PC <- 001

RDM <- [REM] ; RDM <- 11110000

B) Decodificação

RI <- RDM ; RI <- 11110000

Decodificação

C) Busca de Operandos

REM <- PC ; REM <- 001

Read

PC <- PC +1 ; PC <- 010

RDM <- [REM] ; RDM <- 00000011

D)Execução

Y <- RDM

SP <- SP -1 ; SP <- 110

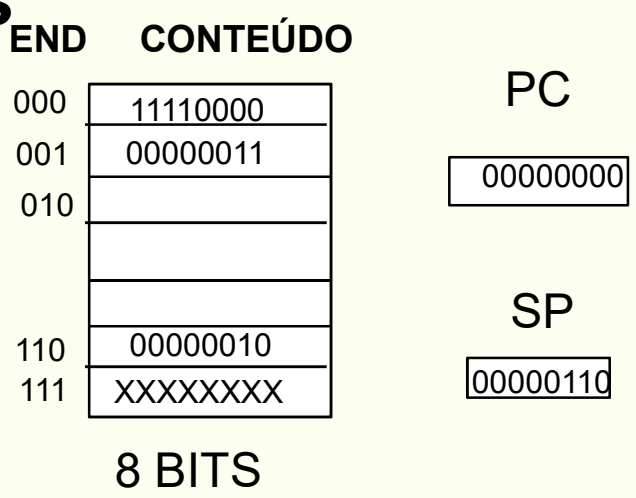
REM <- SP ; REM <- 110

RDM <- PC ; RDM <- 00000010

Write

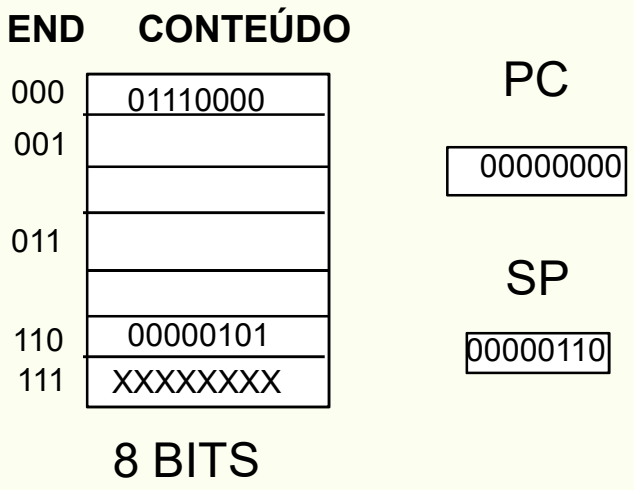
[REM] <- RDM ; [110] <- 00000010

PC <- Y



Exercícios

11 – RET ; PC <- [SP]; SP ++



Exercícios

11 – RET ; PC <- [SP]; SP ++

A) Fetch da Instrução

REM <- PC ; REM <- 000

Read

PC <- PC +1 ; PC <- 001

RDM <- [REM] ; RDM <- 01110000

B) Decodificação

RI <- RDM ; RI <- 01110000

Decodificação

C) Busca de Operandos

NÃO EXISTE

D)Execução

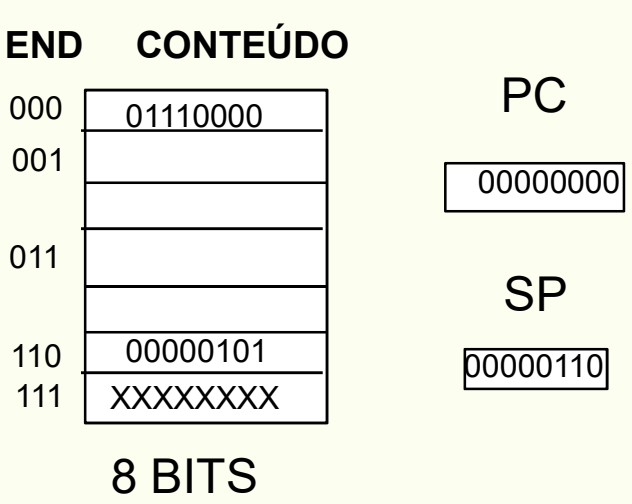
REM <- SP ; REM <- 110

Read

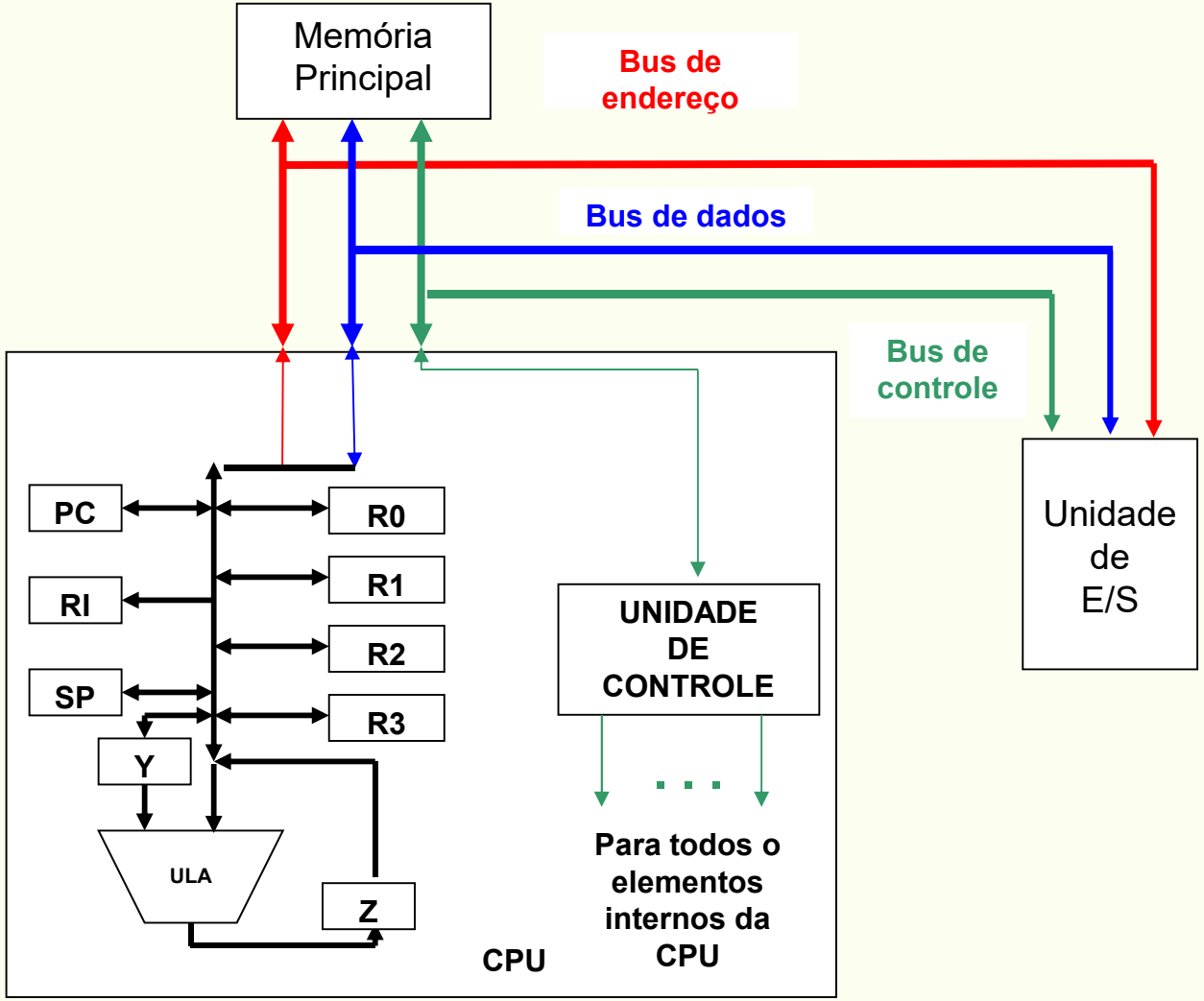
SP <- SP + 1 ; SP <- 111

RDM <- [REM] ; RDM <- [110] ; RDM <- 00000101

PC <- RDM ; PC <- 00000101



ESTUDO DE CASO 2 - CPU HIPOTÉTICA



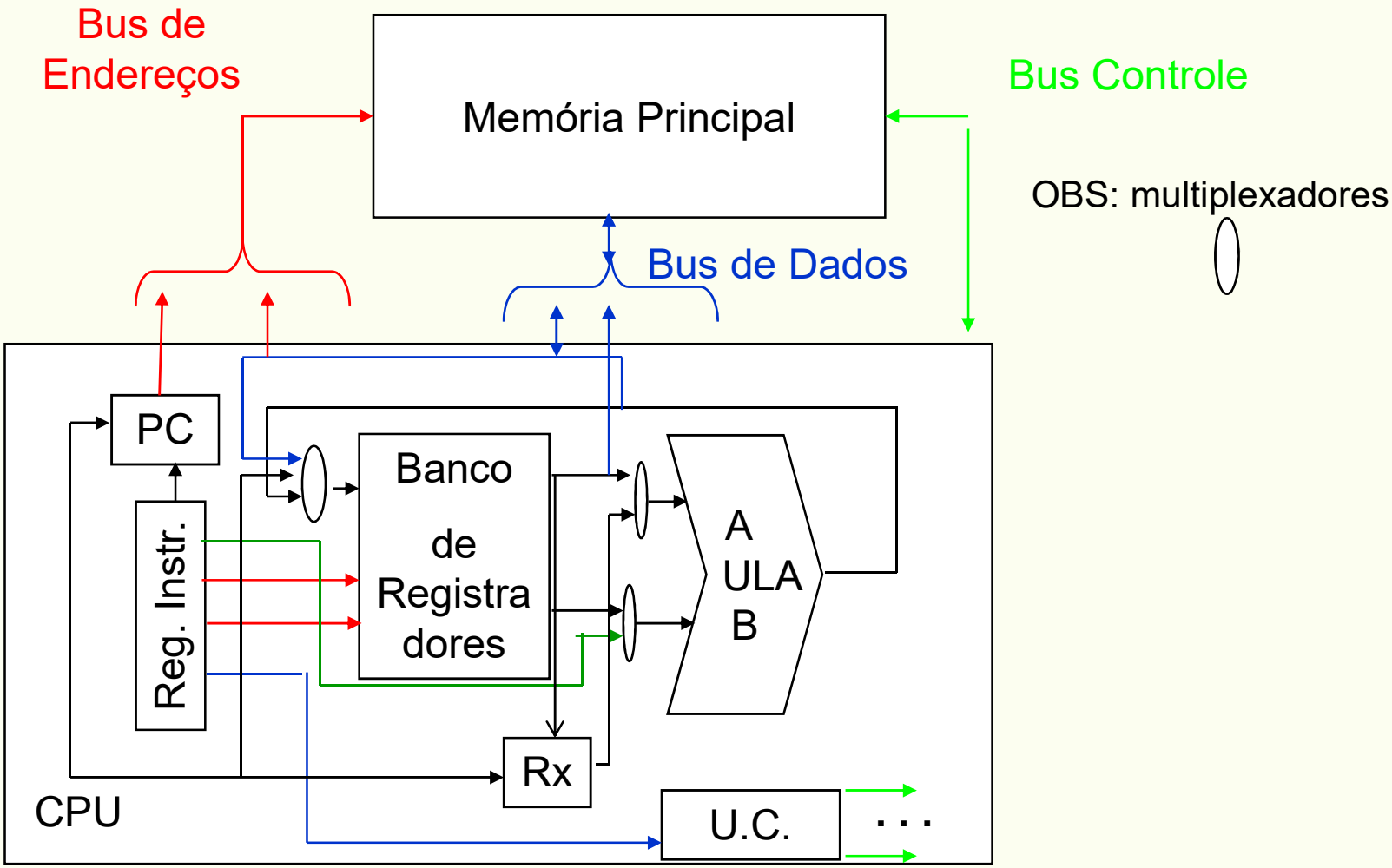
ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Mnemônico	Operandos	Opcode	Significado
Instruções de Movimentação de Dados			
MOV	Reg1,Reg2	0000	Reg1 ←Reg2
MOV	Reg,imed	1000	Reg ← imed
MOV	Reg,[end]	1001	Reg ← MEM[end]
MOV	[end],Reg	1010	MEM[end] ← Reg
Instruções Aritméticas e Lógicas			
ADD	Reg1,Reg2	0001	Reg1 ← Reg1 + Reg2
ADD	Reg,imed	1011	Reg ← Reg + imed
SUB	Reg1,Reg2	0010	Reg1 ← Reg1 - Reg2
SUB	Reg,imed	1100	Reg ← Reg – imed
AND	Reg1,Reg2	0011	Reg1 ← Reg1 *e* Reg2
AND	Reg,imed	1101	Reg ← Reg *e* imed
OR	Reg1,Reg2	0100	Reg1 ← Reg1 *ou* Reg2
Instruções de Manipulação de Pilha			
PUSH	Reg	0101	SP-- , MEM[SP] ← Reg
POP	Reg	0110	Reg←MEM[SP], SP++
Instruções de Controle de Fluxo de Execução			
JMP	end	1110	PC ← end
CALL	end	1111	SP-- , MEM[SP]←PC , PC←end
RET	---	0111	PC← MEM[SP] , SP++

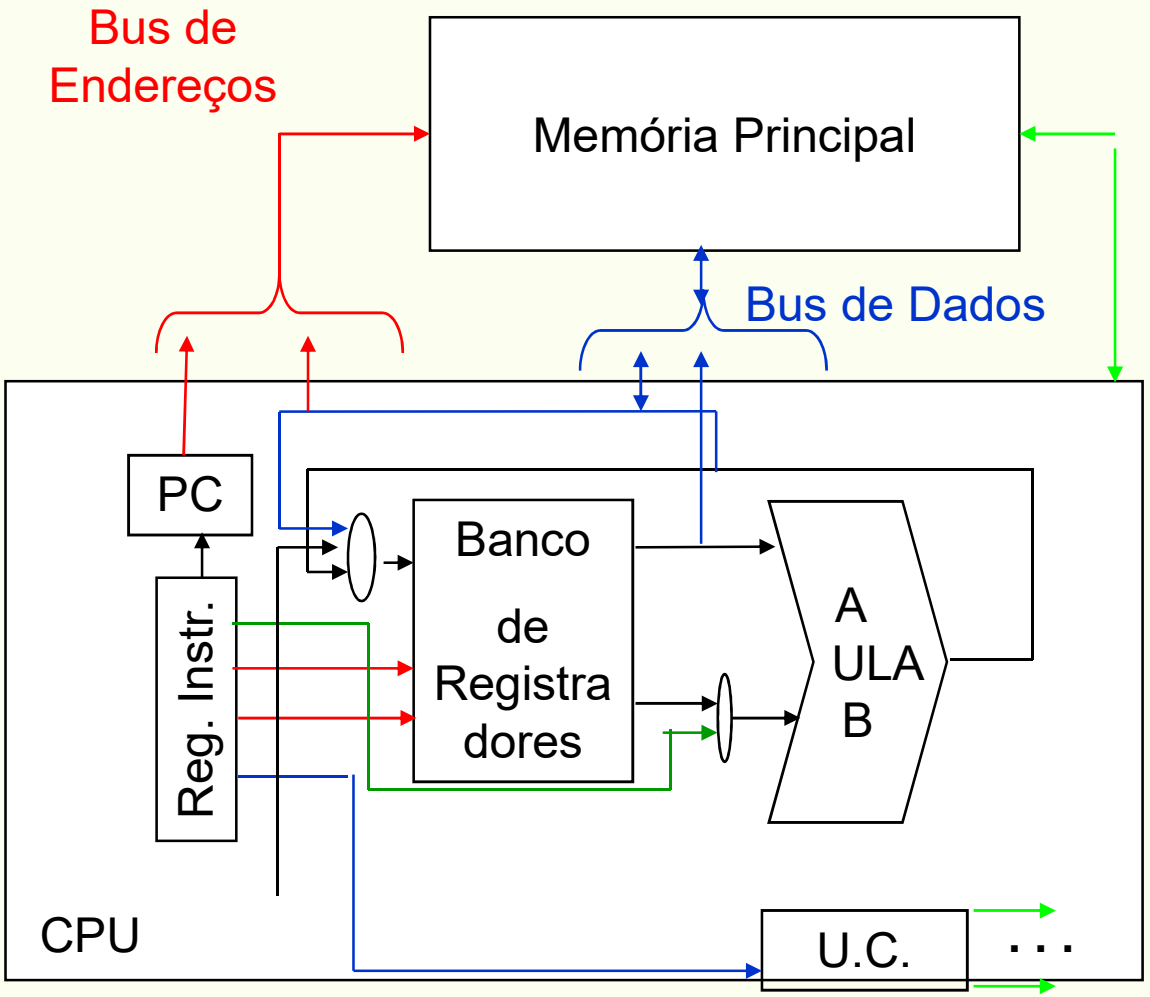
Exercícios

- **Mostrar o ciclo de execução de instruções para todas as instruções do ISA da CPU Hipotética 2.**

CPU HIPOTÉTICA 3



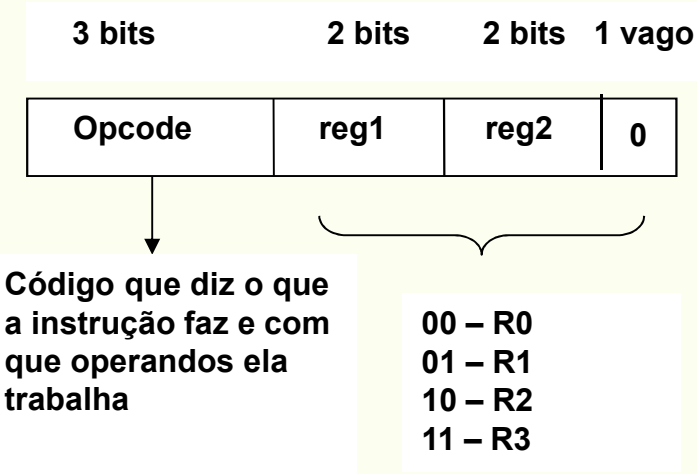
ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM



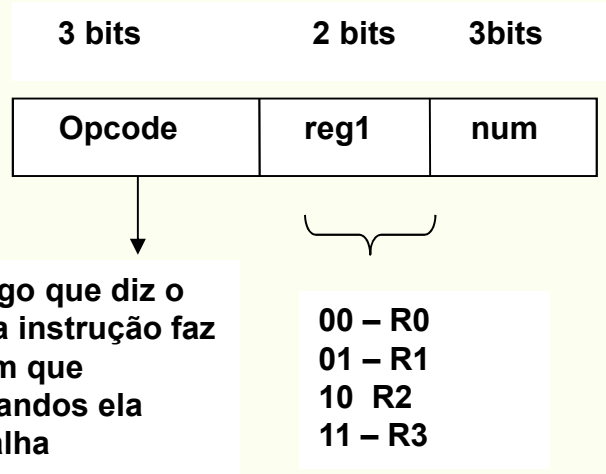
ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

ESTUDO DE CASO - CPU HIPOTÉTICA 3 – Formato de Instruções

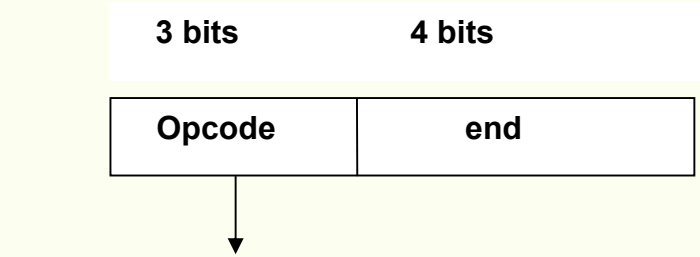
– Formato tipo R – Registrador



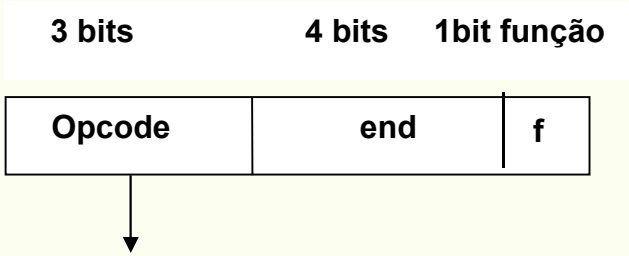
– Formato I – Imediato



– Formato tipo J – Jump



– Formato tipo S - subrotina



ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM

Conjunto de Instruções – CPU Hipotética 3

Mnemônico	Operandos	Opcode	Significado
		Instrução especial	
MV	Rx,Reg	000	Rx ← Reg
		Instruções de load e store	
LW	Reg,num	001	Reg ← MEM[Rx + num]
SW	Reg,num	010	MEM[Rx + num] ← Reg
Instruções Aritméticas e Lógicas			
ADD	Reg1,Reg2	011	Reg1 ← Reg1 + Reg2
SUB	Reg1,Reg2	100	Reg1 ← Reg1 - Reg2
AND	Reg1,Reg2	101	Reg1 ← Reg1 *e* Reg2
		Instruções de Controle de Fluxo de Execução	
JMP	end	110	PC ← end
JAL	end	111 0	Rx ← PC , PC ← end
RET	---	111 1	PC←Rx

Exercícios

- **Mostrar o ciclo de execução de instruções para todas as instruções do ISA da CPU Hipotética 3.**

ORGANIZAÇÃO BÁSICA DE COMPUTADORES E LINGUAGEM DE MONTAGEM