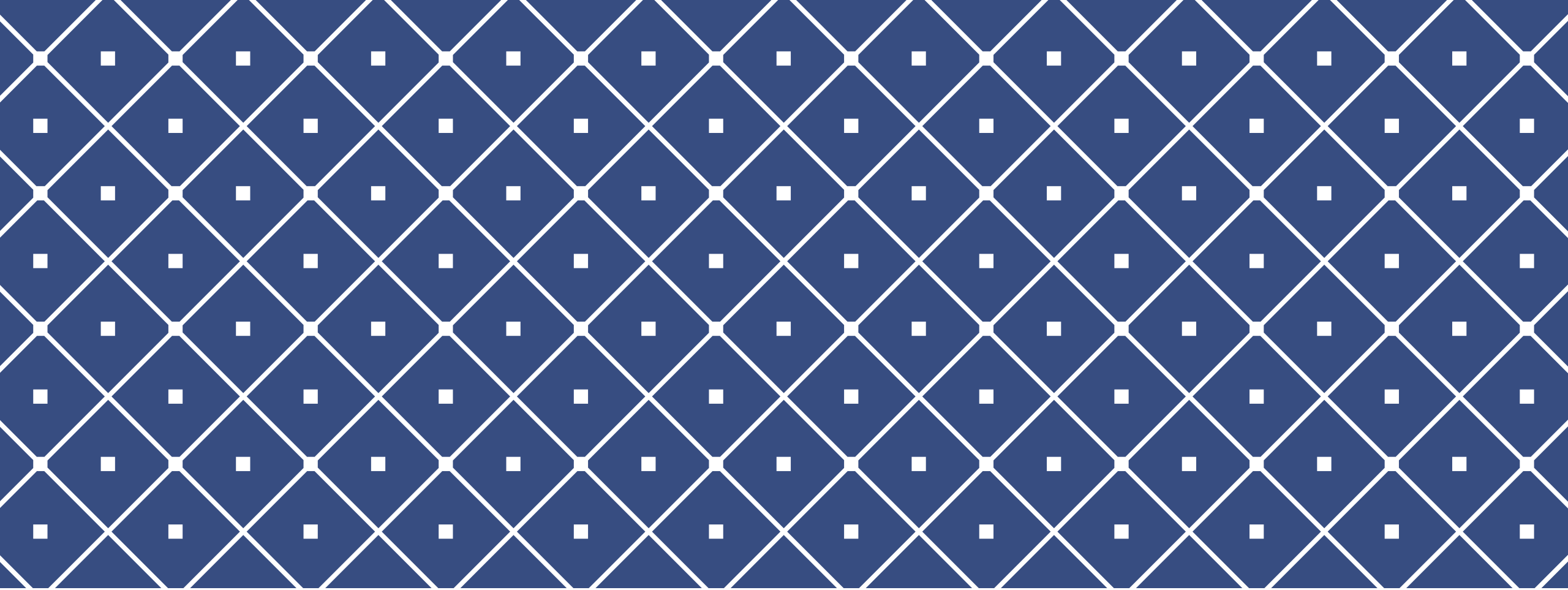


ARQUITETURA HIPOTÉTICA Z70

Prof. André Gustavo Adami

INTRODUÇÃO

A arquitetura hipotética Z70 é uma versão rudimentar da arquitetura x86 da Intel

É uma arquitetura 8 bits com unidade de controle microprogramada

As instruções podem ser codificadas em até 2 bytes

A arquitetura permite endereçar até 256 bytes de memória

Os dados são representados em Complemento de 2

ARQUITETURA HIPOTÉTICA Z70

Barramento

- Endereços/Dados: 8 bits

ULA de 8 bits

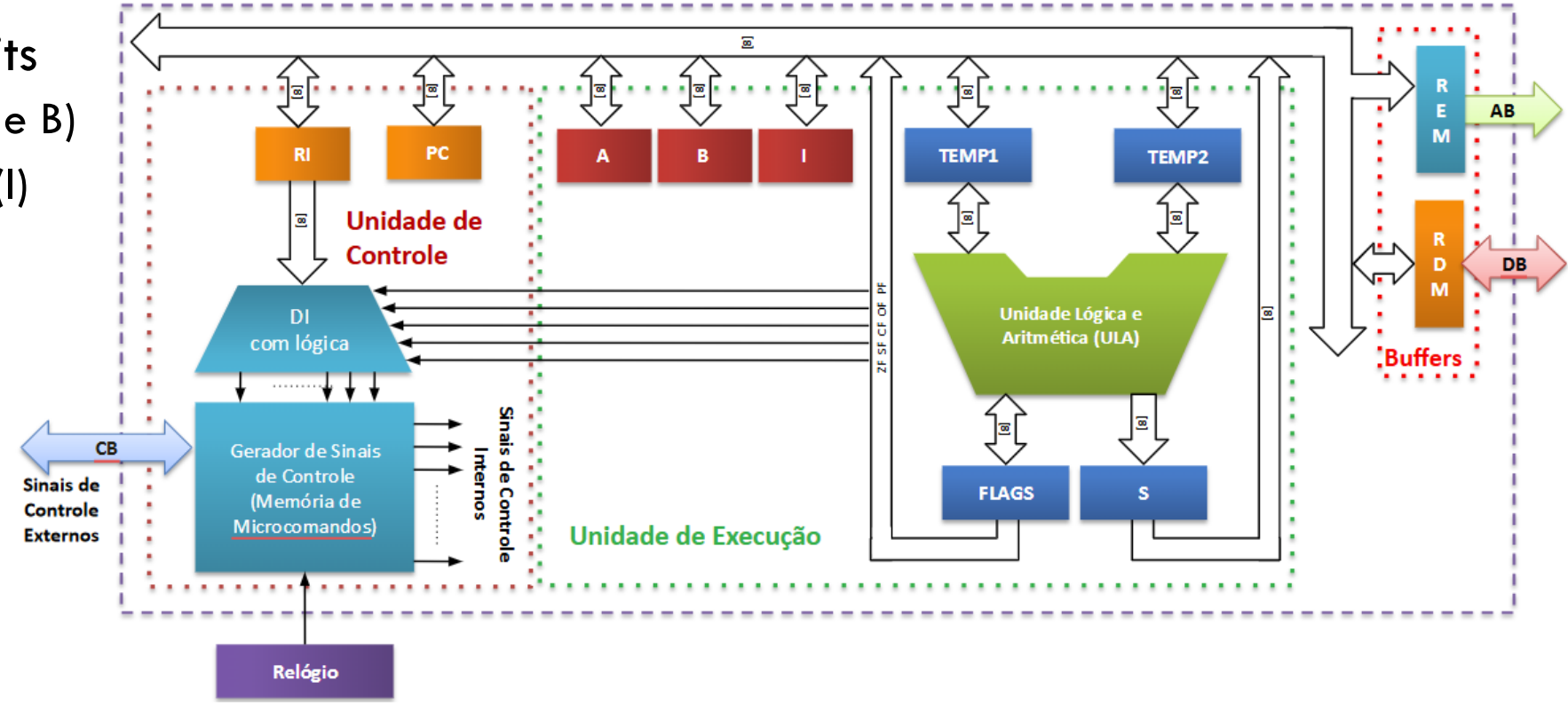
- Registradores auxiliares (TEMP1, TEMP2, S)

Registradores de 8 bits

- 2 de propósito geral (A e B)
- 1 para endereçamento (I)
- 1 de estado (Flags)

Registradores de Controle

- RDM, REM, PC, RI



ARQUITETURA HIPOTÉTICA Z70

Registrador de Estado: Flags

- Registrador de estado alterado pela execução de um instrução (relacionada a ULA)
- O valor de um flag pode ser utilizado para controlar o fluxo do programa

			OF	CF	ZF	PF	SF
7	6	5	4	3	2	1	0

- **CF (carry)**: é ***setado*** (recebe valor 1) se uma operação de adição resulta em "carry" (vai-um) ou se uma operação de subtração resulta em "borrow" (vem-um).
- **PF (parity)**: é ***setado*** quando o resultado de uma operação contém um número par de bits no estado 1.
- **ZF (zero)**: é ***setado*** quando o resultado de uma operação é zero.
- **SF (sign)**: é ***setado*** quando o resultado possuir sinal negativo (copia o bit de sinal).
- **OF (overflow)**: é ***setado*** quando o resultado de uma operação excede os limites de tamanho do operando.

INSTRUÇÕES Z70

As instruções podem possuir um ou dois operandos

<código de operação> <operando>

```
inc A
```

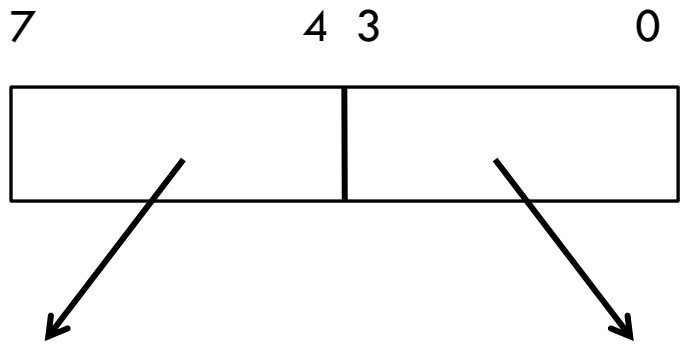
ou

<código de operação> <operando1>,<operando2>

```
add A, B
```

INSTRUÇÕES Z70

O código de operação das instruções é codificado em uma palavra de 8 bits com dois campos



função da instrução (adição, subtração, comparação, desvio, etc.)

operandos (registrador **A**/**B**/**I**, constante, memória) – **modo de endereçamento**

MODOS DE ENDEREÇAMENTO Z70

Imediato

- Um dos operandos é uma constante

Via Registrador

- Todos os operandos são registradores

Direto

- Um dos operandos é um endereço de memória (variável escalar)

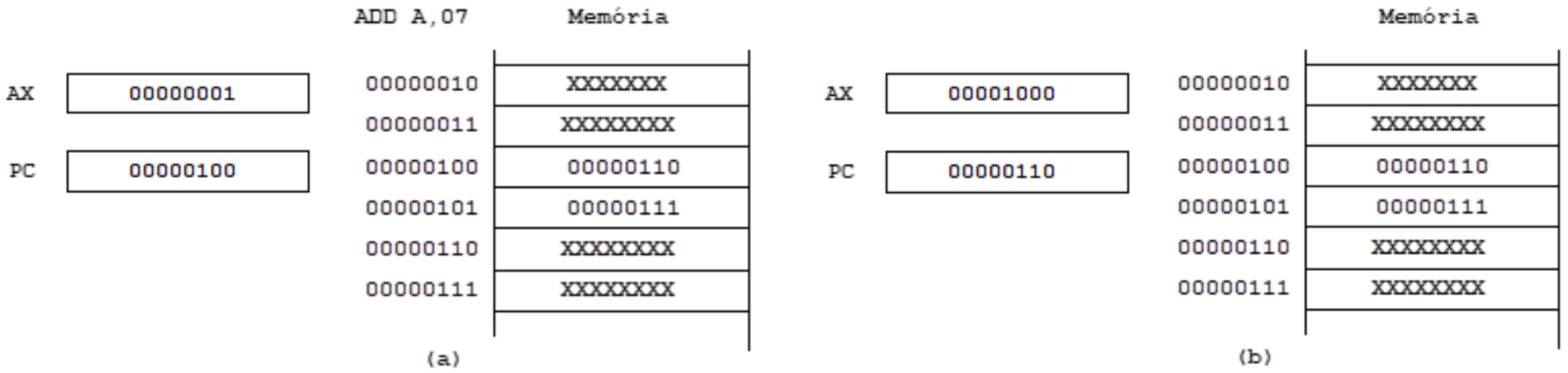
Indireto

- Um dos operandos é um registrador que possui um endereço de memória a ser acessado

Modo de Endereçamento	Codificação (bits 3, 2, 1, 0)	Operandos
Via Registrador	0H	A, B
	1H	B, A
	2H	A, I
	3H	I, A
Indireto	4H	A, [I]
	5H	[I], A
Imediato	6H	A, constante
	7H	B, constante
	8H	I, constante
Indireto	9H	[I], constante
Direto	AH	A, [memória]
	BH	B, [memória]
	CH	[memória], A
	DH	[memória], B

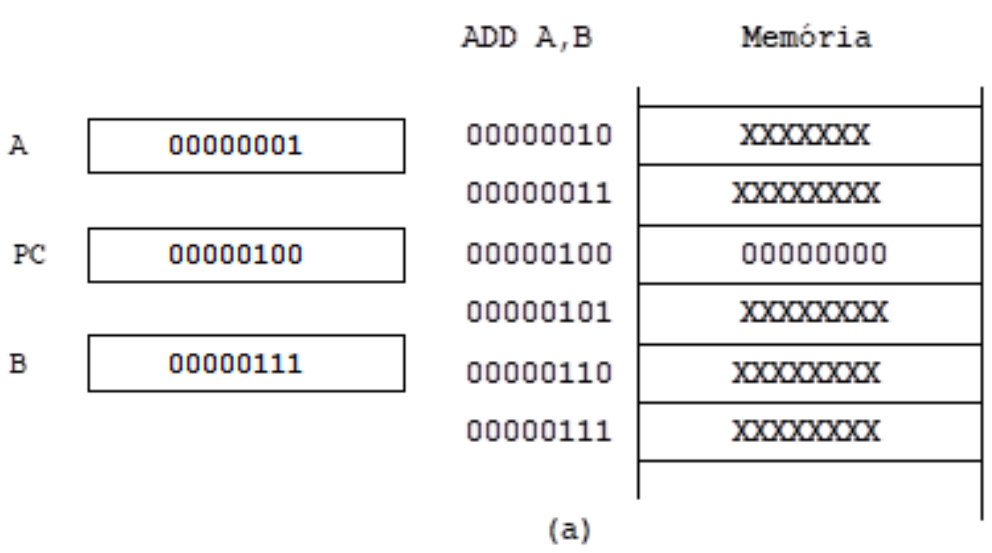
MODOS DE ENDEREÇAMENTO: IMEDIATO

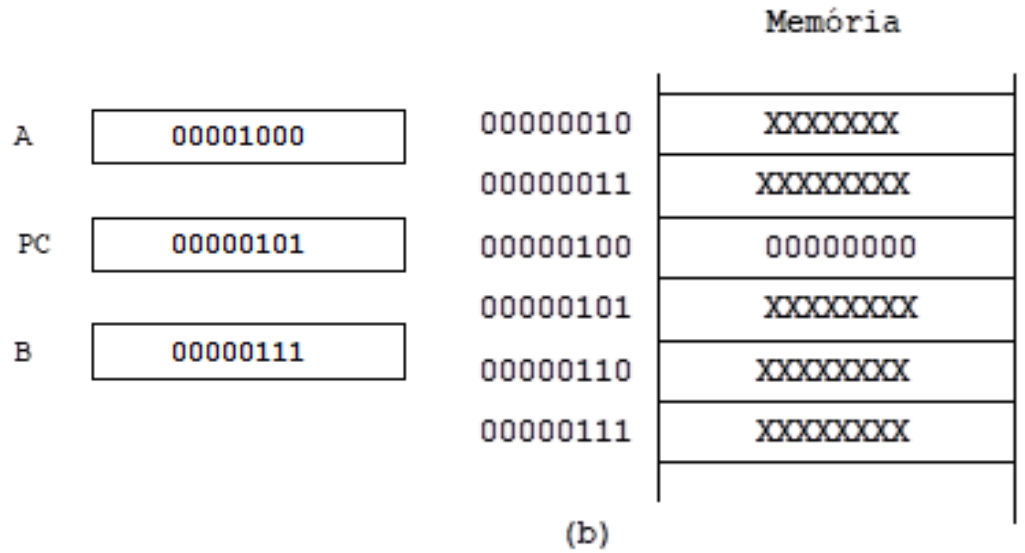
O operando é uma constante numérica que segue o código da instrução

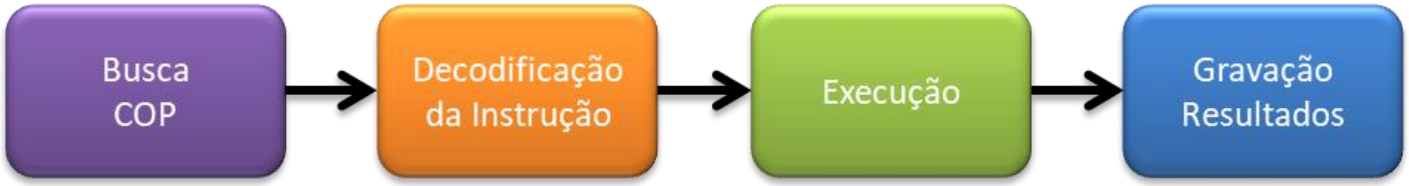


MODOS DE ENDEREÇAMENTO: VIA REGISTRADOR

O(s) operando(s) é(são) o(s) conteúdo(s) de um registrador(es)

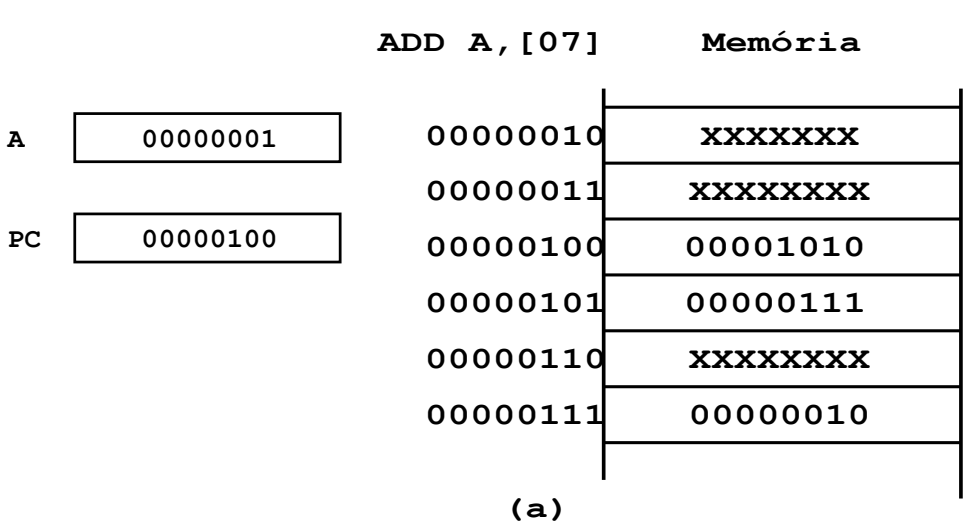


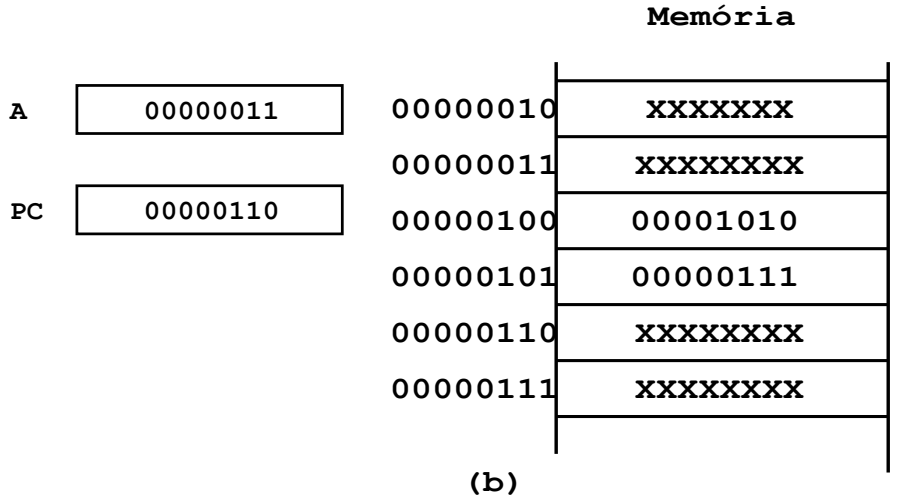


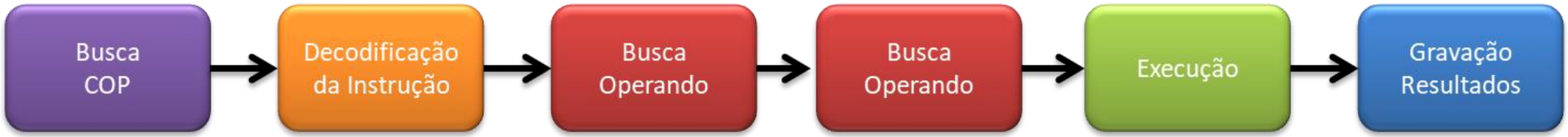


MODOS DE ENDEREÇAMENTO: DIRETO

O operando localiza-se em um endereço especificado na instrução (imediatamente após o código de operação)

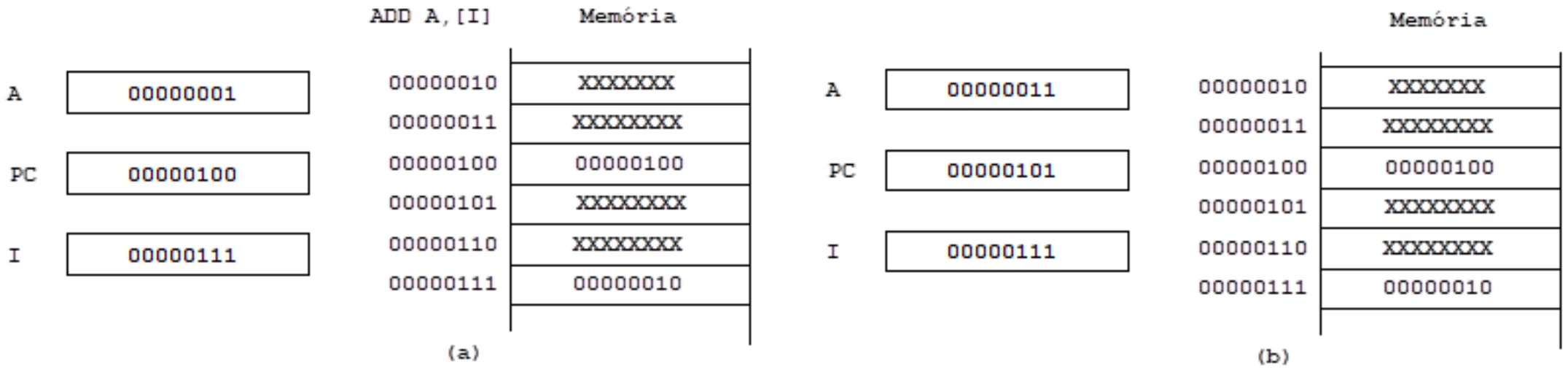






MODOS DE ENDEREÇAMENTO: INDIRETO

O operando é o conteúdo de uma posição de memória apontada pelo registrador **I**



INSTRUÇÕES Z70

Tipo de Instrução	Codificação (bits 7, 6, 5, 4)	Mnemônico	Função
Aritmética	0H	add	Adição
	1H	sub	Subtração
	2H	cmp	Comparação
	3H	inc	Incremento
	4H	dec	Decremento
Lógica	5H	and	E Lógico
	6H	or	Ou Lógico
	7H	not	Não Lógico
	8H	shr	Deslocamento para Direita
	9H	shl	Deslocamento para Esquerda
Desvio	AH	jmp	Desvio Incondicional (COP = $A0_{16}$)
	AH	jz	Desvio Se Zero (COP = $A1_{16}$)
	AH	js	Desvio Se Negativo (COP = $A2_{16}$)
	AH	jc	Desvio Se Vai-um (COP = $A3_{16}$)
	AH	jo	Desvio Se Estouro (COP = $A4_{16}$)
	AH	jp	Desvio Se Paridade (COP = $A5_{16}$)
Movimentação	BH	mov	Movimentação
Controle	FH	nop	Faz nada (ciclos de clock ociosos - COP = FF_{16})

INSTRUÇÕES Z70

As instruções de incremento (**inc**), decremento (**dec**), não lógico (**not**), deslocamento para a direita (**shr**) e deslocamento para a esquerda (**shl**) permitem somente os modos de endereçamento via registrador (isto é, A, B, ou I) e indireto (isto é, [I])

Modo de Endereçamento	Codificação (bits 3, 2, 1, 0)	Operando
Via Registrador	0H	A
	1H	B
	2H	I
Indireto	4H	[I]

Para diferenciar um desvio condicional do outro, o campo para o modo de endereçamento é utilizado para identificar o tipo de desvio

- Somente o modo de endereçamento imediato é permitido para as instruções de desvio, pois elas possuem somente um operando que é um endereço de memória

INSTRUÇÕES Z70: ARITMÉTICAS

add op1,op2	soma os conteúdos de op1 e op2; coloca o resultado em op1
sub op1,op2	subtrai de op1 o conteúdo de op2
cmp op1,op2	compara os valores de op1 e op2. O resultado da comparação é a ativação ou não dos "flags" de **F**
inc op	incrementa op de uma unidade
dec op	decrementa op de uma unidade

Flags afetados: CF, OF, ZF, PF, SF

INSTRUÇÕES Z70: ARITMÉTICAS

add A,[200];	Soma ao reg A o conteúdo da posição 200
add A,8;	Soma ao reg A o valor 8
sub A,B;	Subtrai o valor em reg B do reg A
sub A,[I];	Subtrai do reg A o conteúdo da posição apontada por I
cmp A,8;	Compara A com 8
inc A;	Soma 1 a A
dec A;	Subtrai 1 de A

INSTRUÇÕES Z70: LÓGICAS

and op1, op2	realiza o e lógico "bit" a "bit" de op1 e op2; coloca o resultado em op1
or op1, op2	realiza o ou lógico "bit" a "bit" de op1 e op2
not op	realiza a complementação "bit" a "bit" de op
shr op	desloca o conteúdo de op um "bit" para a direita (para o carry flag)
shl op	desloca o conteúdo de op um "bit" para a esquerda (para o carry flag)

Flags afetados

- and, or, not: ZF, PF, SF
- shr e shl: CF, OF, ZF, PF, SF

INSTRUÇÕES Z70: LÓGICAS

and A, 0; Zera o reg A

or A, FFH; Seta todos os bits (A = FFH)

not A; Se A=0, A terá o valor 11111111_2 (FFH)

shl A; Se A=FFH, CF=1 e A terá o valor 11111110_2 (FEH)

shr A; Se A=FFH, CF=1 e A terá o valor 01111111_2 (7FH)

INSTRUÇÕES Z70: DESVIO

jmp rótulo desvia incondicionalmente para um rótulo de endereço

jz rótulo se ZF= 1, desvia para um rótulo de endereço

js rótulo se SF= 1, desvia para um rótulo de endereço

jc rótulo se CF= 1, desvia para um rótulo de endereço

jo rótulo se OF= 1, desvia para um rótulo de endereço

jp rótulo se PF= 1, desvia para um rótulo de endereço

Flags afetados: nenhum

INSTRUÇÕES Z70: DESVIO

O rótulo é um identificador de uma instrução do programa

O rótulo é sempre seguido por ":" a fim de que o mesmo seja identificado como rótulo

```
LACO: ADD A, B
```

O rótulo `LACO` identifica a linha da instrução `ADD A, B`.

Não é necessário que a instrução esteja na mesma linha que o rótulo (pode estar na próxima linha)

```
LACO:
    ADD A, B
```

Não existe rótulo sem uma instrução, pois os rótulos são substituídos pelo endereço da instrução que a identificam

INSTRUÇÕES Z70: DESVIO

and A,0;	Zera A e ZF = 1
jz LACO;	Como ZF = 1, desvia para o endereço do rótulo LACO
cmp A,8;	Compara A com 8
js FIM;	Desvia para o endereço do rótulo FIM, se A < 8
jmp LACO;	Desvia incondicionalmente para o endereço do rótulo LACO

INSTRUÇÕES Z70: MOVIMENTAÇÃO

mov op1, op2 realiza a movimentação do dado em op2 para op1. As instruções de movimentação têm sempre ao menos um registrador como operando (fonte ou destino)

Flags afetados: nenhum

Exemplos

```
mov A, 0;      A = 0, i.e., zera A
mov B,[200];   Coloca em B o conteúdo do endereço 200
mov A, I;      A = B
```

CICLO DE INSTRUÇÃO

A execução das instruções segue um fluxo dentro da arquitetura desde a busca da instrução até a gravação do resultado

O fluxo é controlado pela unidade de controle através da geração de sinais para os componentes da arquitetura

A programação de baixo nível demanda que o programador conheça os sinais que são gerados pela unidade de controle a fim de produzir códigos mais eficientes

CICLO DE INSTRUÇÃO

Instrução: add A, B

- Registrador A
  - Conteúdo: 03H
- Registrador B
  - Conteúdo: 04H
- Instrução add
  - Código de operação: 00H
  - Endereço: 10H

CICLO DE INSTRUÇÃO

Instrução: add A, B

1.	REM ← PC	REM ← 10H	1ª Fase – Busca do COP
2.	RDM ← MEM[REM]	RDM ← 00H	
3.	RI ← RDM	RI ← 00H	
4.	PC ← PC + 1	PC ← 11H	
5.	RI → DI		2ª Fase – Decodificação
6.	TEMP1 ← A	TEMP1 ← 03H	4ª Fase – Execução da Instrução
7.	TEMP2 ← B	TEMP2 ← 04H	
8.	S ← TEMP1 + TEMP2	S ← 07H	
9.	A ← S	A ← 07H	5ª Fase – Gravação do Resultado

CICLO DE INSTRUÇÃO

Instrução: add [80H], A

- Mem[80H]
  - Conteúdo: 20H
- Registrador A
  - Conteúdo: 07H
- Instrução add
  - Código de operação: 0CH
  - Endereço: 11H

CICLO DE INSTRUÇÃO

Instrução: add [80H], A

1.	REM ← PC	REM ← 11H	1ª Fase – Busca do COP
2.	RDM ← MEM[REM]	RDM ← 0CH	
3.	RI ← RDM	RI ← 0CH	
4.	PC ← PC + 1	PC ← 12H	
5.	RI → DI		2ª Fase – Decodificação
6.	REM ← PC	REM ← 12H	3ª Fase – Busca de Operando
7.	RDM ← MEM[REM]	RDM ← 80H	
8.	PC ← PC + 1	PC ← 13H	
9.	REM ← RDM	REM ← 80H	
10.	RDM ← MEM[REM]	RDM ← 20H	
11.	TEMP1 ← RDM	TEMP1 ← 20H	4ª Fase – Execução da Instrução
12.	TEMP2 ← A	TEMP2 ← 07H	
13.	S ← TEMP1 + TEMP2	S ← 27H	
14.	RDM ← S	RDM ← 27H	5ª Fase – Gravação do Resultado
15.	MEM[REM] ← RDM	MEM[80H] ← 27H	

PROGRAMAÇÃO Z70

Um programa que soma o conteúdo de todas as posições de memória entre 0 e 3 e escreve o resultado na posição 0

```
mov A,[0]
mov B,[1]
add A,B
mov B,[2]
add A,B
mov B,[3]
add A,B
mov [0],A
```

OU

```
mov A, [0]
add A, [1]
add A, [2]
add A, [3]
mov [0],A
```

OU

```
        mov A,[0]
        mov I,1
Soma:   add A,[I]
        inc I
        cmp I,4
        jz Fim
        jmp Soma
Fim:    mov [0],A
```

OU

```
        mov I,3
        mov A,0
Soma:   add A,[I]
        dec I
        jz Fim
        jmp Soma
Fim:    add [0],A
```

PROGRAMAÇÃO Z70: EXERCÍCIO 1

Faça um programa que realize o produto de dois números armazenados nos registradores A e B

- Assuma que o programa começa com as seguintes instruções

```
mov A, 5
mov B, 4
```

- Faça também a codificação do programa em hexadecimal apontando o endereço de cada instrução

PROGRAMAÇÃO Z70: EXERCÍCIO 1

End	Codificação	Programa
00H	B6H 05H	mov A, 5
02H	B7H 04H	mov B, 4
04H	B3H	mov I, A
05H	41H	LACO:dec B
06H	A1H 0BH	jz FIM
08H	02H	add A,I
09H	A0H 05H	jmp LACO
0BH	FFH	FIM: nop

PROGRAMAÇÃO Z70: EXERCÍCIO 2

Um programa que, dentro de um laço, coloca no registrador A as oito primeiras potências de 2 (1, 2, 4, 8, 16, 32, 64 e 128)

```
End       Codificação    Programa
00H       B6H 01H                mov A,1
02H       90H            LACO:   shl A
03H       A1H 07H                jo FIM
05H       A0H 02H                jmp LACO
07H       FFH            FIM:    nop
```

PROGRAMAÇÃO Z70: EXERCÍCIO 3

Um programa que, dentro de um laço, coloca no registrador **A** as oito primeiras potências de 2 (1, 2, 4, 8, 16, 32, 64 e 128) sem deixar que o registrador mude de valor em 128

```
End       Codificação    Programa
00H       B6H 01H                mov A,1
02H       B7H 00H                mov B,0
04H       90H            LACO:   shl A
05H       31H                    inc B
06H       27H 07H                cmp B,7
08H       A2H 04H                js LACO; B-7<0?Vá p/LACO
```

PROGRAMAÇÃO Z70: EXERCÍCIO 2 (VERSÃO 2)

Um programa que, dentro de um laço, coloca no registrador **A** as oito primeiras potências de 2 (1, 2, 4, 8, 16, 32, 64 e 128) sem deixar que o registrador mude de valor em 128

```
End       Codificação    Programa
00H       B6H 01H                mov A, 1
02H       90H            LACO:   shl A
03H       26H 80H                cmp A, 128
05H       A1H 09H                jz FIM
07H       A0H 02H                jmp LACO
09H       FFH            FIM:    nop
```