

CONJUNTO DE INSTRUÇÕES DE UM PROCESSADOR (UCP)

LINGUAGENS

Constituída de sequência de zeros (0) e uns (1)

Linguagem Máquina

```
1010 11001
1011 11010
1100 11011
```

Linguagem *Assembly*

```
LOD Y
ADD Z
STR X
```

Cada instrução em ASSEMBLY constitui-se em um mnemônico (uma forma fácil de se lembra) de uma instrução em linguagem de máquina.

Exemplo:

```
LOD ~ 101011001
ADD ~ 101111010
STR ~ 110011011
```

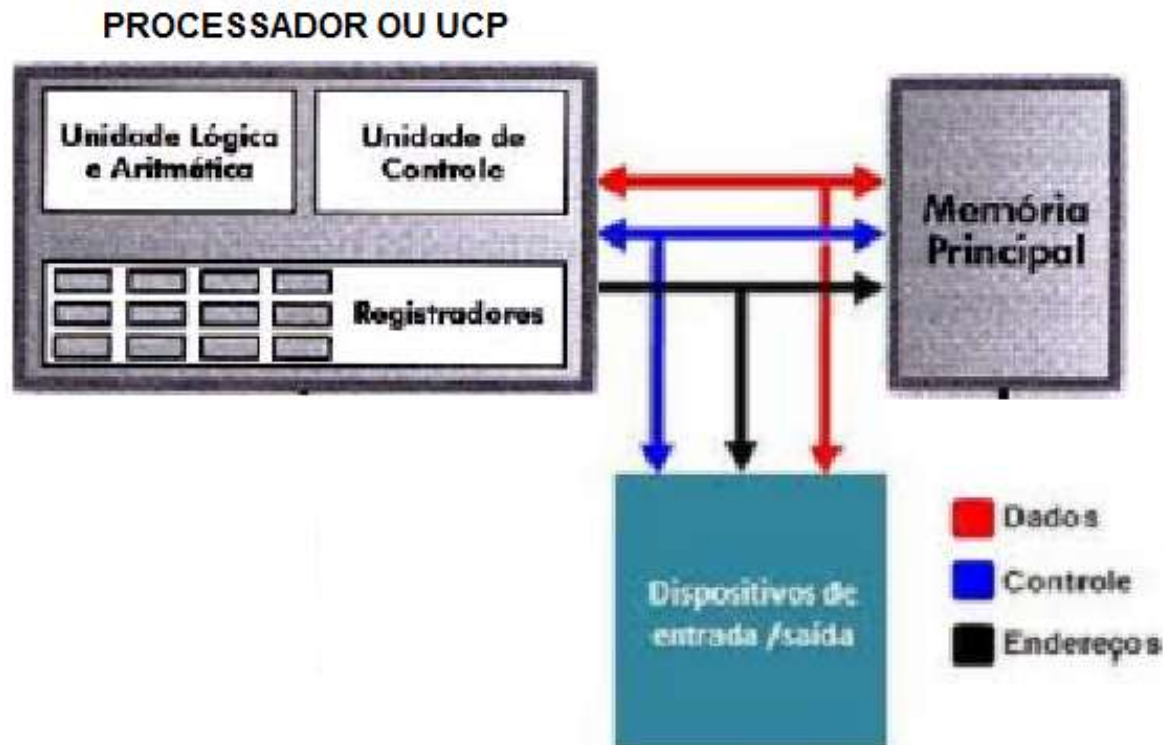
Linguagem de Alto Nivel (C)

```
X=Y+Z
```

É uma linguagem cujas instruções são de fácil entendimento do usuário.

PROGRAMAÇÃO EM ASSEMBLY

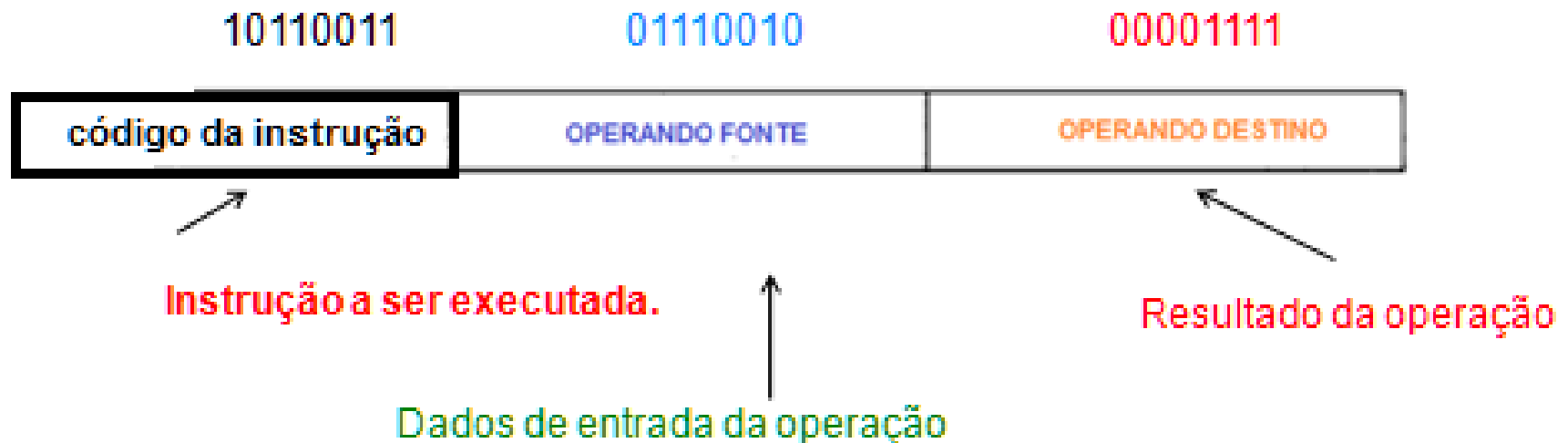
- Para programar em **ASSEMBLY** o programador deve conhecer:
 - O conjunto de registradores do processador (UCP)
 - A estrutura da memória principal (MP)
 - Tipos de dados (endereços, números, caracteres, etc.) disponíveis diretamente na máquina
 - Funcionamento da ULA (unidade lógica aritmética)



ELEMENTOS DE UMA INSTRUÇÃO

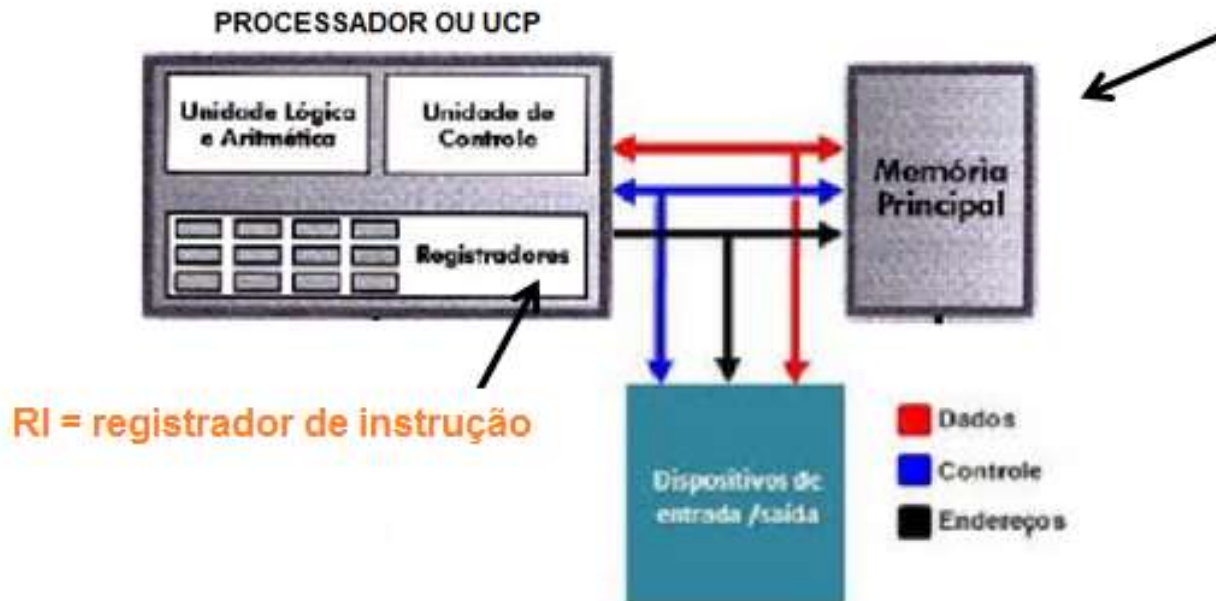
- Código da instrução
- Operando fonte
- Operando de destino
- Endereço da próxima instrução (normalmente a próxima instrução da memória)

instrução simples



EXECUÇÃO DE UMA INSTRUÇÃO

- A instrução é lida da memória e vai para o registrador de instrução (RI) do PROCESSADOR.
- O **PROCESSADOR** extrai os dados dos vários campos da instrução armazenadas no RI (registrador de instrução) e efetua a operação requerida.

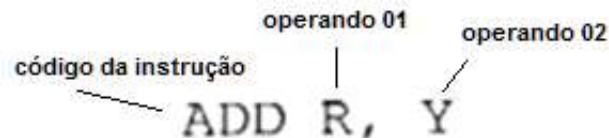


INSTRUÇÃO EM ASSEMBLY

- É uma representação simbólica para cada instrução em linguagem de máquina.
- Exemplos

ADD	Adição
SUB	Subtração
MPY	Multiplicação
DIV	Divisão
LOAD	Carregar dados da memória
STOR	Armazenar dados na memória

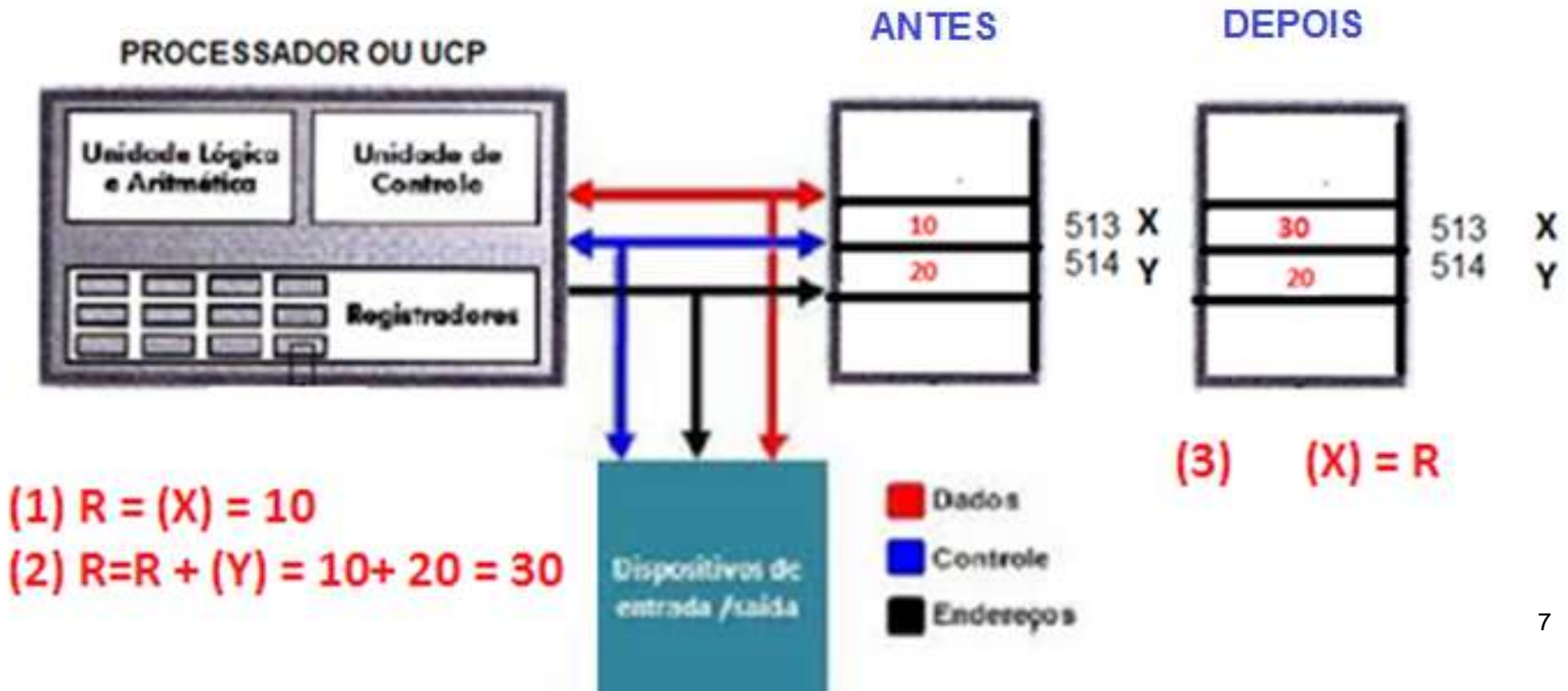
- Os operandos são também representados de maneira simbólica.



ADICIONAR O VALOR CONTIDO NA POSIÇÃO Y COM CONTEÚDO DO REGISTRADOR R

INSTRUÇÃO LING ALTO NÍVEL → INSTRUÇÃO EM ASSEMBLY

- Uma instrução em LING ALTO NÍVEL (tal como C), requer uma ou várias instruções em ASSEMBLY:
 - $X = X + Y$ (instrução em linguagem de alto-nível)
 - Supondo que as variáveis X em Y correspondam as posições de memória 513 e 514. A instrução acima pode ser implementado **com três instruções em ASSEMBLY**:
 - (1) Carregar um registrador com o conteúdo de posição de memória 513.
 - (2) Adicionar o conteúdo da posição de memória 514 ao conteúdo do registrador.
 - (3). Armazenar o conteúdo do registrador na posição de memória 513



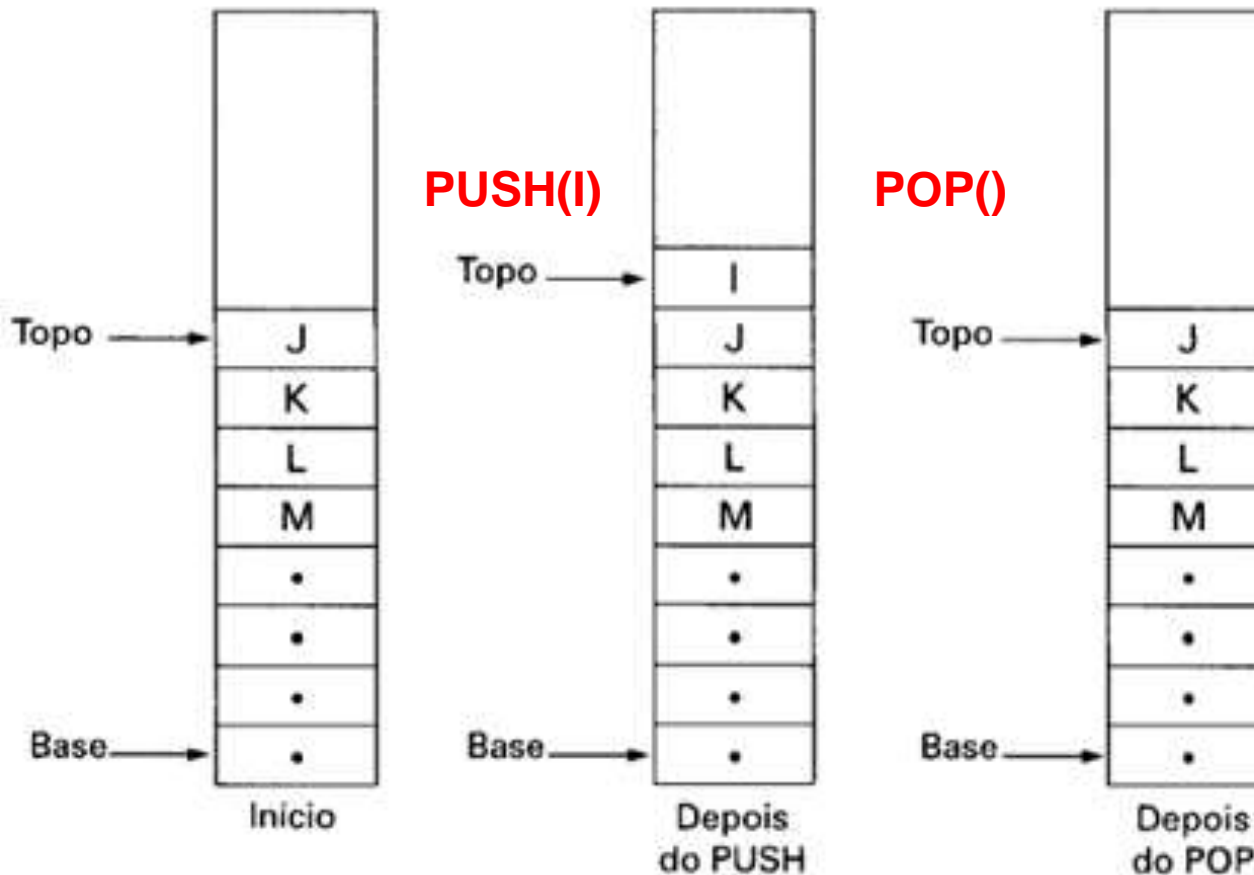
EXEMPLO DE PROGRAMA EM ASSEMBLY

<u>Operação</u>		<u>Operando</u>	
LDA	~ CARREGAR	I	
ADD		J	
ADD	~ ADICIONAR	K	-
STA	~ ARMAZENAR	N	

COMPUTAÇÃO DE $N = I + J + K$

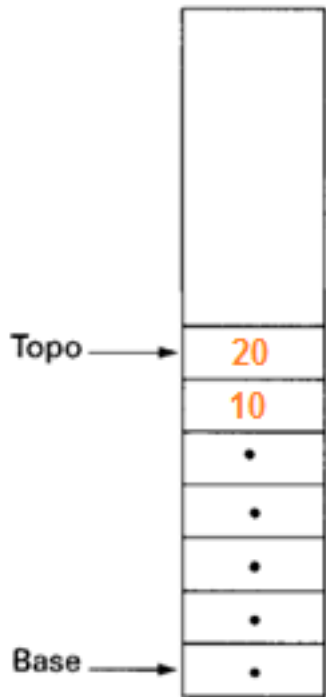
PILHA

- Conjunto ordenado onde a inserção (PUSH) ou remoção (POP) de um ITEM é feita pelo TOPO da PILHA.
 - PUSH (item): adiciona elemento na PILHA
 - POP: remove o item do topo da PILHA



USO DE PILHAS (INSTRUÇÕES BINÁRIAS)

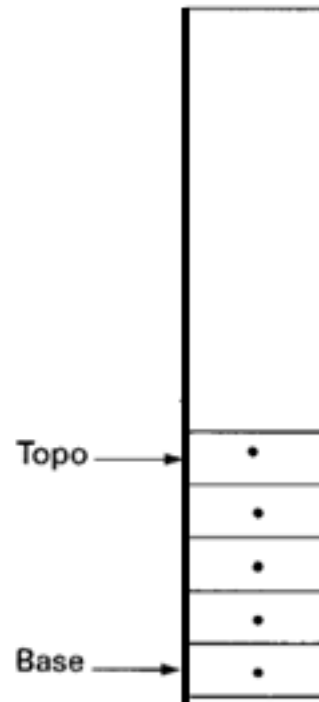
- Pilhas são usadas para implementar operações binárias ou unárias
 - Instruções binárias:
 - Retiram dois operandos do topo da pilha e colocam o resultado na pilha.
 - Exemplo: multiplicação, divisão, soma, subtração.
 - **A = POP; B = POP; C = A+B; PUSH (C).**



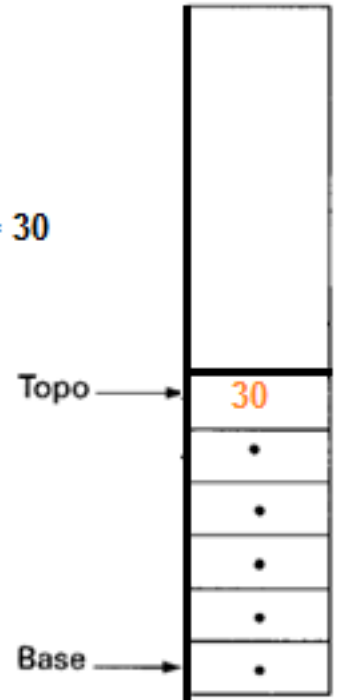
A = POP () = 20



B = POP () = 10



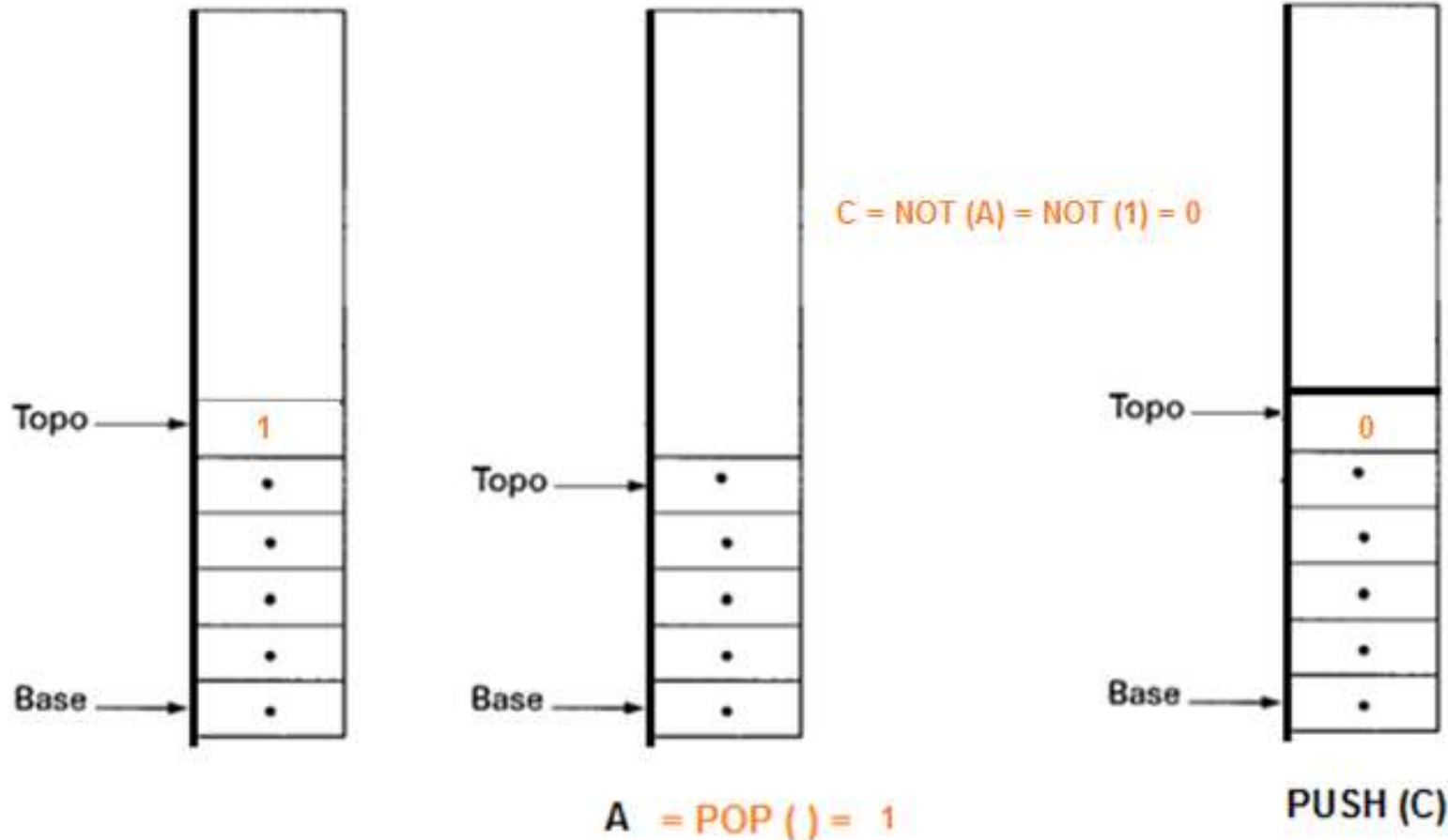
$$C = A + B = 30$$



PUSH (C)

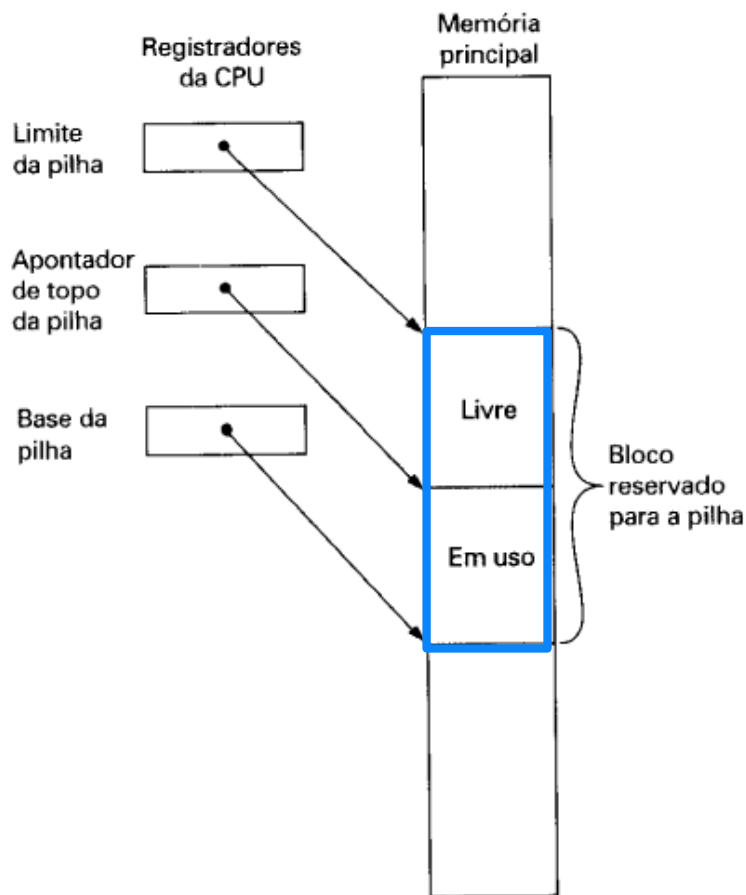
USO DE PILHAS (INSTRUÇÕES UNÁRIAS)

- Pilhas são usadas para implementar operações binárias ou unárias
 - Instruções unárias:
 - requerem um operando (NOT por exemplo), usa o item do topo da pilha.
 - Exemplo: operação NOT
 - **A = POP; C= NOT(C) ; PUSH(C)**



IMPLEMENTAÇÃO DE UM PILHA

- A PILHA é um bloco de posições de memória principal
 - Base da pilha: endereço da posição inicial do bloco de memória reservado para a pilha.
 - Limite da pilha: endereço da extremidade do bloco de memória reservado para a pilha.
 - Apontador da pilha: endereço do topo da pilha.



NÚMERO DE OPERANDOS DE UMA INSTRUÇÃO

- INSTRUÇÕES COM ZERO (0), UM (1) , DOIS (2) OU TRÊS (3) OPERANDOS

Número de endereços	Representação simbólica	Interpretação
3	OP A, B, C	$A \leftarrow B \text{ OP } C$
2	OP A, B	$A \leftarrow A \text{ OP } B$
1	OP A	$AC \leftarrow AC \text{ OP } A$
0	OP	$T \leftarrow (T-1) \text{ OP } T$

AC = acumulador (registrador especial do processador)

T = topo da pilha

A, B, C = registrador ou posição de memória

**QUANTO MAIS OPERANDOS NAS INSTRUÇÕES DE UM PROCESSADOR
MAIS COMPLEXO É O PROCESSADOR**

NÚMERO DE OPERANDOS DE UMA INSTRUÇÃO

■ Instruções com 03 operandos:

- Dois endereços de operandos e um endereço de resultado.

Instrução		Comentário
SUB	Y, A, B	$Y \leftarrow A - B$
MPY	T, D, E	$T \leftarrow D \times E$
ADD	T, T, C	$T \leftarrow T + C$
DIV	Y, Y, T	$Y \leftarrow Y \div T$

■ Instruções com 02 operandos:

- Um endereço de operando e um endereço de resultado.

Instrução		Comentário
MOVE	Y, A	$Y \leftarrow A$
SUB	Y, B	$Y \leftarrow Y - B$
MOVE	T, D	$T \leftarrow D$
MPY	T, E	$T \leftarrow T \times E$
ADD	T, C	$T \leftarrow T + C$
DIV	Y, T	$Y \leftarrow Y \div T$

NÚMERO DE OPERANDOS EM UMA INSTRUÇÃO

■ Instruções com 01 operando:

- Usa um endereço implícito (o do registrador acumulador (AC))

Instrução		Comentário
LOAD	D	$AC \leftarrow D$
MPY	E	$AC \leftarrow AC \times E$
ADD	C	$AC \leftarrow AC + C$
STOR	Y	$Y \leftarrow AC$
LOAD	A	$AC \leftarrow A$
SUB	B	$AC \leftarrow AC - B$
DIV	Y	$AC \leftarrow AC \div Y$
STOR	Y	$Y \leftarrow AC$

PROJETO DO CONJUNTO DE INSTRUÇÕES

- TIPO DE OPERANDO
- TIPO DE INSTRUÇÃO
- TIPO DE DADOS

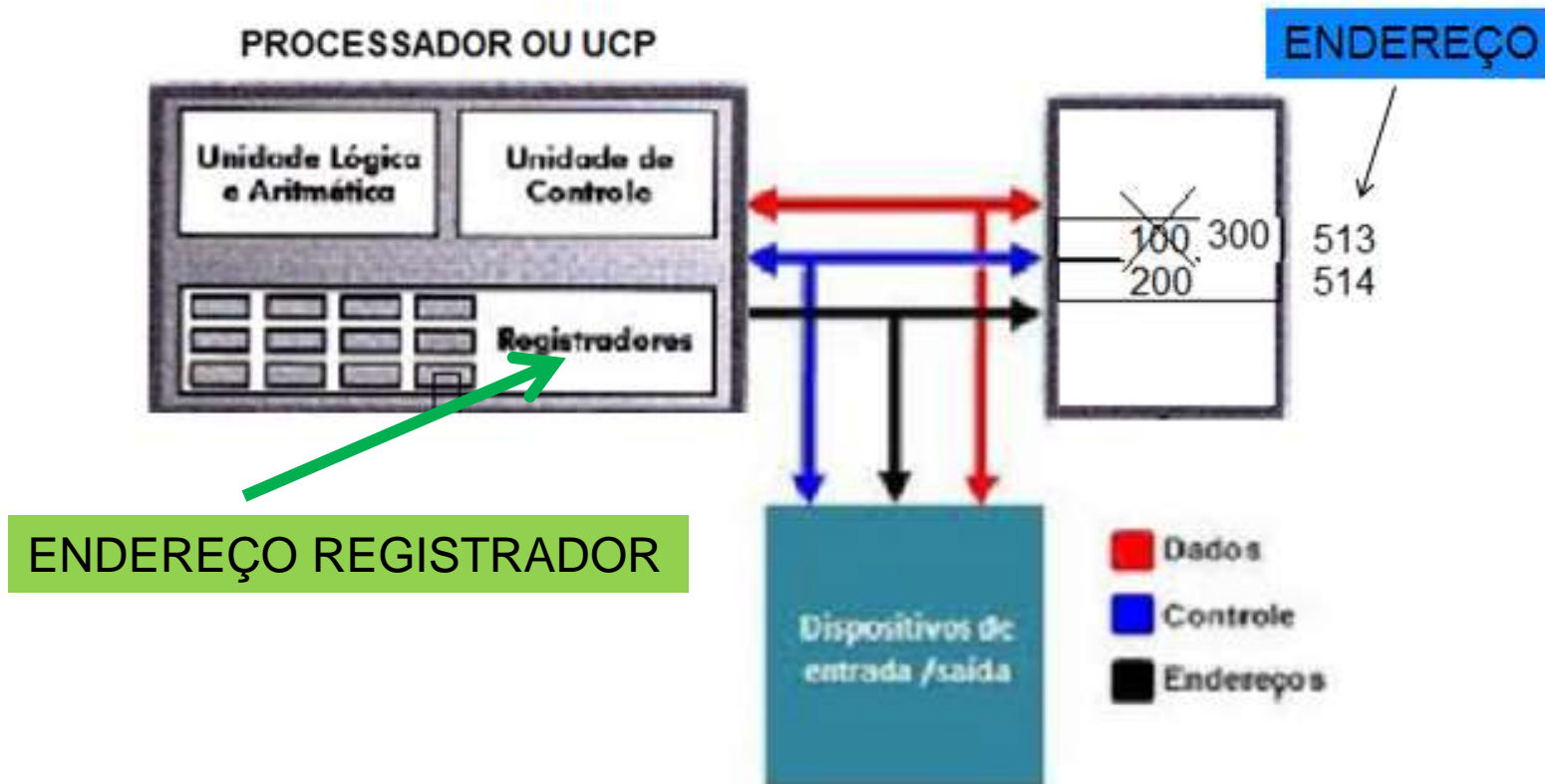
TIPOS DE OPERANDOS DE UMA INSTRUÇÃO

- Instruções operam sobre operandos. Os tipos de operandos são
 - ENDEREÇOS
 - NÚMEROS
 - CARACTERES
 - ETC

TIPOS DE OPERANDOS DE UMA INSTRUÇÃO

■ ENDEREÇO

- Exemplo: endereço de uma posição de memória, endereço de um registrador



TIPOS DE OPERANDOS DE UMA INSTRUÇÃO

■ NÚMERO

- Exemplos: inteiro, ponto flutuante.

■ CARACTERE

- O código mais usado é o ASCII. É um código binário que codifica um conjunto de 128 sinais: 95 sinais gráficos (letras do alfabeto latino, sinais de pontuação e sinais matemáticos) e 33 sinais de controle.

Bin	Oct	Dec	Hex	Sinal
0010 0000	040	32	20	(espaço)
0010 0001	041	33	21	!
0010 0010	042	34	22	"
0010 0011	043	35	23	#
0010 0100	044	36	24	\$
0010 0101	045	37	25	%
0010 0110	046	38	26	&
0010 0111	047	39	27	'
0010 1000	050	40	28	(
0010 1001	051	41	29)
0010 1010	052	42	2A	*

Bin	Oct	Dec	Hex	Sinal
0100 0000	100	64	40	@
0100 0001	101	65	41	A
0100 0010	102	66	42	B
0100 0011	103	67	43	C
0100 0100	104	68	44	D
0100 0101	105	69	45	E
0100 0110	106	70	46	F
0100 0111	107	71	47	G
0100 1000	110	72	48	H
0100 1001	111	73	49	I
0100 1010	112	74	4A	J

Bin	Oct	Dec	Hex	Sinal
0110 0000	140	96	60	`
0110 0001	141	97	61	a
0110 0010	142	98	62	b
0110 0011	143	99	63	c
0110 0100	144	100	64	d
0110 0101	145	101	65	e
0110 0110	146	102	66	f
0110 0111	147	103	67	g
0110 1000	150	104	68	h
0110 1001	151	105	69	i
0110 1010	152	106	6A	j

TIPO DE INSTRUÇÃO

- As CLASSE DE INSTRUÇÕES abaixo são encontradas nos diversos tipos de PROCESSADORES:
 - 1. Instruções de transferência de dados
 - 2. Instruções aritméticas
 - 3. Instruções lógicas
 - 4. Instruções de conversão
 - 5. Instruções de E/S
 - 6. Instruções de controle do sistema
 - 7. Instruções de transferência de controle.

INSTRUÇÃO DE TRANSFERÊNCIA DE DADOS

- Deve especificar os endereços fonte e destino (**posição de memória, registrador ou topo da pilha**)

Tipo	Nome da operação	Descrição
Operações de transferência de dados	Move	<u>Transfere</u> uma palavra ou <u>bloco da fonte</u> para o <u>destino</u>
	Store	<u>Transfere</u> uma palavra do <u>processador</u> para a <u>memória</u>
	Load	<u>Transfere</u> uma palavra da <u>memória</u> para o <u>processador</u>
	Exchange	<u>Troca</u> os conteúdos dos <u>operandos fonte e de destino</u>
	Clear	<u>Transfere</u> uma <u>palavra contendo 0s</u> para o <u>destino</u>
	Set	<u>Transfere</u> um <u>palavra contendo 1s</u> para o <u>destino</u>
	Push	<u>Transfere</u> uma palavra <u>da fonte</u> para o <u>topo da pilha</u>
	Pop	<u>Transfere</u> uma palavra <u>do topo da pilha</u> para o <u>destino</u>

INSTRUÇÃO ARITMÉTICA

- A maioria dos processadores oferece instruções como **adição, subtração, multiplicação, divisão, incremento, etc.**

Tipo	Nome da operação	Descrição
Operações aritméticas	Add	<u>Soma dois operandos</u>
	Subtract	Calcula a <u>diferença</u> entre <u>dois operandos</u>
	Multiply	Calcula o <u>produto</u> de <u>dois operandos</u>
	Divide	Calcula o <u>quociente</u> de <u>dois operandos</u>
	Absolute	<u>Substitui</u> o <u>operando</u> pelo seu <u>valor absoluto</u>
	Negate	<u>Muda o sinal</u> do <u>operando</u>
	Increment	<u>Soma 1</u> ao <u>operando</u>
	Decrement	<u>Subtrai 1</u> do <u>operando</u>

INSTRUÇÃO LÓGICA

- São efetuadas sobre dados binários :

P	Q	NOT P	P AND Q	P OR Q	P XOR Q	P=Q
0	0	1	0	0	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	1	0	1	1	0	1

- Podem ser aplicadas **bit a bit**. Por exemplo: se dois registradores contem os dados

(R1) = 10100101

(R2) = 00001111

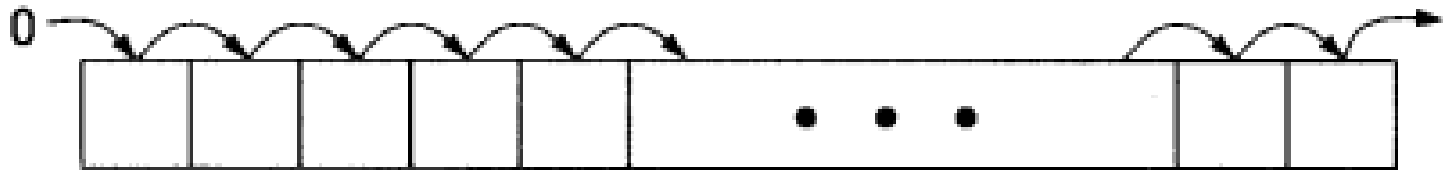
então,

(R1) AND (R2) = 00000101

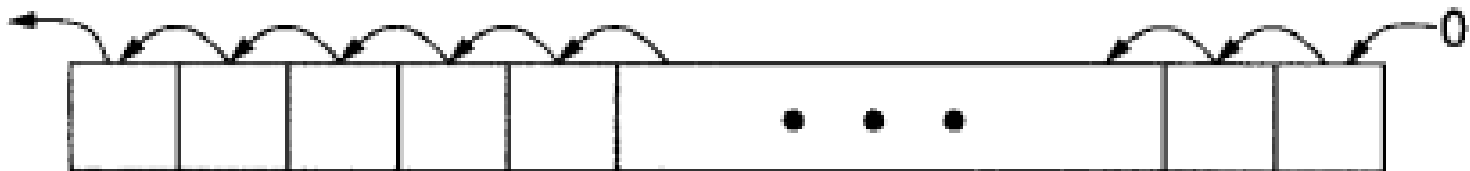
INSTRUÇÃO LÓGICA (deslocamento lógico)

■ Exemplo de Uso:

- Transmissão de caracteres de dados para um dispositivo de E/S (um caracteres de cada vez).



(a) Deslocamento lógico para a direita

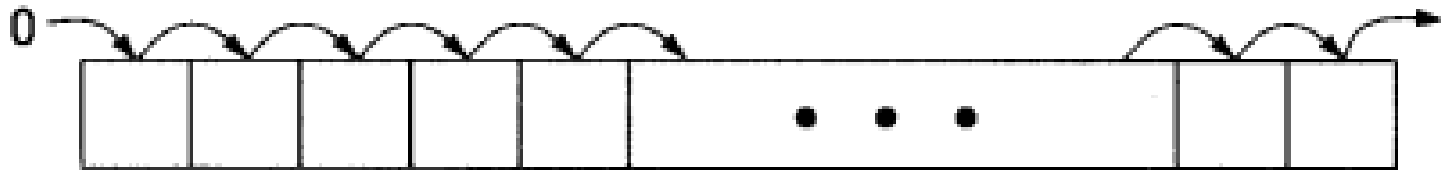


(b) Deslocamento lógico para a esquerda

INSTRUÇÃO LÓGICA (deslocamento lógico)

- Um deslocamento para **esquerda** ou para **direita** corresponde, respectivamente, à **multiplicação** ou **divisão** por 2.

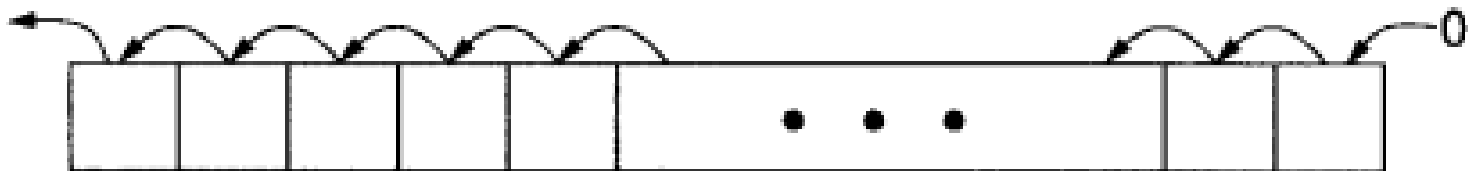
0110 → 6



(a) Deslocamento lógico para a direita

DIVISÃO

0011 → 3



(b) Deslocamento lógico para a esquerda

MULTIPLICAÇÃO



1110 → 12

INSTRUÇÃO LÓGICA (rotação)

- Preserva todos os bits sobre as quais uma instrução é efetuada



INSTRUÇÕES LÓGICAS (rotação)

Tipo	Nome da operação	Descrição
	AND OR NOT (Complemento) Exclusive-OR	Efetua a operação lógica especificada, bit a bit
Operações lógicas	Test	<u>Testa a condição especificada; atualiza códigos de condição (flags), de acordo com o resultado</u>
	Compare	Efetua uma <u>comparação lógica ou aritmética de dois ou mais operandos; atualiza códigos de condição (flags), de acordo com o resultado</u>
	Set control variables	Classe de instruções para especificar informação de controle, para fins de proteção, tratamento de interrupção, controle de temporização etc.
	Shift	<u>Deslocamento de operando para a esquerda (direita), introduzindo constantes no final</u>
	Rotate	<u>Rotação circular de operando para a esquerda (direita)</u>

INSTRUÇÃO DE CONVERSÃO

- Mudam ou operam sobre o formato de dados.
- Exemplo: conversão de um número de decimal para binário.

Tipo	Nome da operação	Descrição
Operações de conversão	Translate	Traduz valores armazenados em uma seção da memória, com base em uma tabela de correspondências
	Convert	Converte o conteúdo de uma palavra de uma representação para outra (por exemplo, decimal empacotado para binário)

INSTRUÇÃO DE ENTRADA E SAÍDA

- Existe uma variedade abordagens:

Tipo	Nome da operação	Descrição
Operações de E/S	Read (input)	Transfere <u>dados da porta ou dispositivo de E/S especificado</u> para o <u>destino</u> (por exemplo, memória principal ou registrador de processador)
	Write (output)	Transfere <u>dados da fonte especificada</u> para uma <u>porta ou um dispositivo de E/S</u>
	Start I/O	Transfere <u>instruções para o processador de E/S</u> , para iniciar uma operação de E/S
	Test I/O	Transfere <u>informação de estado do sistema de E/S</u> para o destino especificado

TIPO DE CONTROLE DO SISTEMA

- São as que só podem ser executadas com processador no modo privilegiado (modo KERNEL)
- São reservadas para uso do sistema operacional

INSTRUÇÃO DE TRANSFERÊNCIA DE CONTROLE

- Alteram a seqüência normal de execução das instruções.
 - Instrução de desvio
 - Instrução de salto
 - Instrução de Chamadas de procedimento.

INSTRUÇÃO DE TRANSFERÊNCIA DE CONTROLE (**DESVIO**)

■ INCONDICIONAL

- BR

■ CONDICIONAL

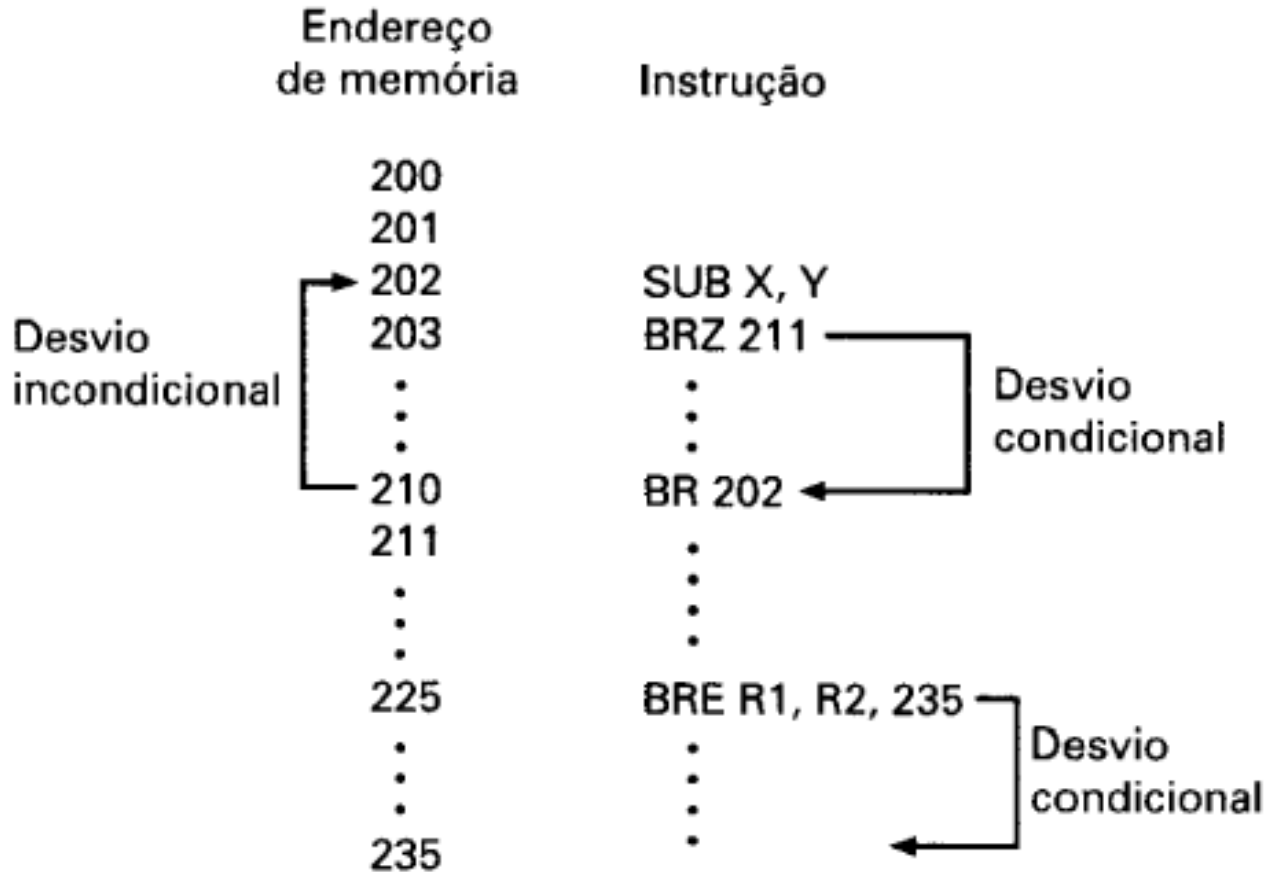
Uma DETERMINADA INSTRUÇÃO pode atualizar um código de condição com os valores 0, POSITIVO, NEGATIVO ou OVERFLOW.

- BRP X = desvia para instrução de endereço X se resultado for POSITIVO
- BRN X = desvia para instrução de endereço X se resultado for NEGATIVO
- BRZ X = desvia para instrução de endereço X se resultado for ZERO
- BRO X = desvia para instrução de endereço X se ocorrer OVERFLOW

– BRE R1,R2,X

- Desvia para instrução de endereço X se conteúdo de R1 = conteúdo de R2
- **Observação → BR vem de BRANCH = DESVIO**

INSTRUÇÃO DE TRANSFERÊNCIA DE CONTROLE (**DESVIO**)



→ NO ENDEREÇO 202 SUBTRAI-SE X de Y

→ NO ENDEREÇO 203 VERIFICA-SE SE O RESULTADO FOI ZERO. EM CASO POSITIVO SALTA PARA ENDEREÇO 211

→ NO ENDEREÇO 210 EXISTE UM DESVIO INCONDICIONAL PARA ENDEREÇO 202

→ NO ENDEREÇO 225 VERIFICA-SE SE R1=R2. EM CASO POSITIVO SALTA PARA O ENDEREÇO 235

INSTRUÇÃO DE TRANSFERÊNCIA DE CONTROLE (**SALTO**)

- Incluem um endereço de desvio implícito

301

•

•

•

309 ISZ R1

ISZ — increment-and-skip-if-zero

310 BR 301

311

ISZ R1

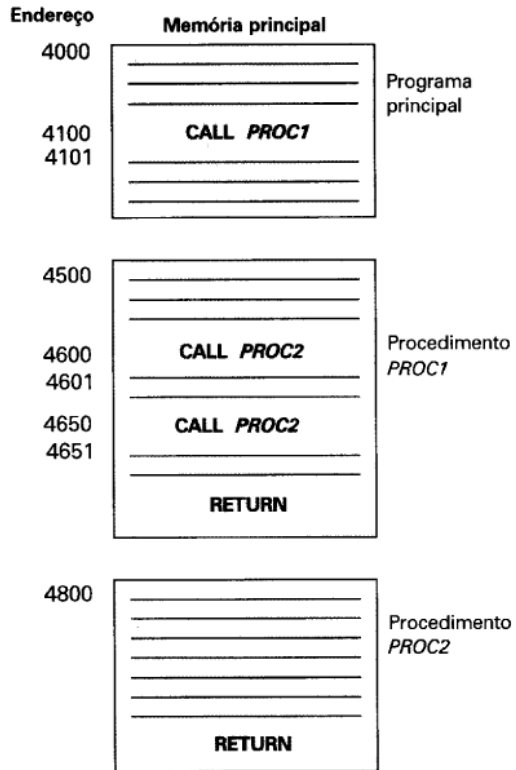
- INCREMENTA R1 e ENQUANTO FOR DIFERENTE DE ZERO VAI PARA ENDEREÇO 310 (QUE POSSUI UM DESVIO INCONDICIONAL)
- CASO CONTRÁRIO VAI PARA ENDEREÇO 311

INSTRUÇÃO DE TRANSFERÊNCIA DE CONTROLE

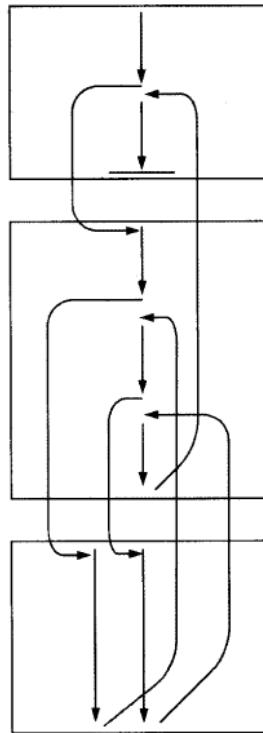
(CHAMADA DE PROCEDIMENTO)

- Envolve uma Instrução de chamada (desvia a instrução corrente para o início do procedimento) e uma instrução de retorno (que provoca o retorno da execução do procedimento para o endereço em que ocorreu a chamada).
- Uso de PILHA é uma abordagem para armazenar o endereço de retorno de uma chamada de procedimento.

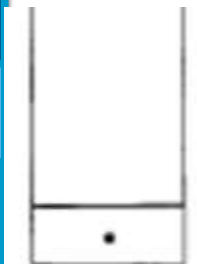
INSTRUÇÃO DE TRANSFERÊNCIA DE CONTROLE (CHAMADA DE PROCEDIMENTO)



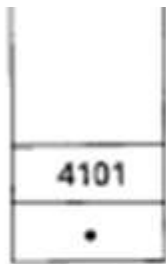
(a) Chamadas e retornos de procedimentos



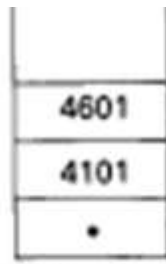
(b) Seqüência de execução



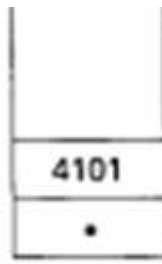
(a) Conteúdo inicial da pilha



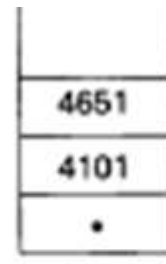
(b) Depois de executar a instrução CALL PROC1



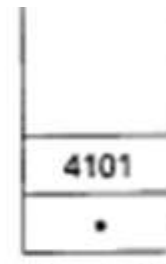
(c) Depois de executar a primeira instrução CALL PROC2



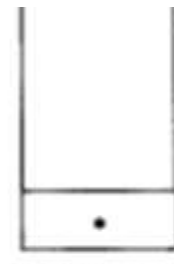
(d) Depois de executar a instrução RETURN



(e) Depois de executar a segunda instrução CALL PROC2



(f) Depois de executar a instrução RETURN



(g) Depois de executar a instrução RETURN

INSTRUÇÃO DE TRANSFERÊNCIA DE CONTROLE

Tipo	Nome da operação	Descrição
Operações de transferência de controle	Jump (branch)	Desvio incondicional; <u>carrega o PC com o endereço especificado</u>
	Jump conditional	<u>Testa a condição</u> especificada; <u>carrega ou não o PC</u> com o endereço especificado, conforme o resultado do teste
	Jump to subroutine	Armazena informação de controle do programa corrente em uma posição conhecida; desvia para o endereço especificado
	Return	Substitui o conteúdo do PC e de outros registradores com os valores armazenados em uma posição conhecida
	Execute	Busca o operando em uma posição especificada e executa o valor desse operando como uma instrução; não modifica o PC
	Skip	Incrementa o PC (para o endereço da próxima instrução)
	Skip conditional	Testa a condição especificada; desvia ou não com base no resultado do teste
	Halt	Pára a execução do programa
	Wait (hold)	Pára a execução do programa; testa a condição especificada repetidamente; retoma a execução quando a condição é satisfeita
	No operation	Não efetua nenhuma operação e continua a execução do programa