Faculdade de Computação e Engenharia Elétrica

Microprocessadores e Microcontroladores

Programação Básica

Prof. Dr. Elton Alves

Instrução de salto incondicional

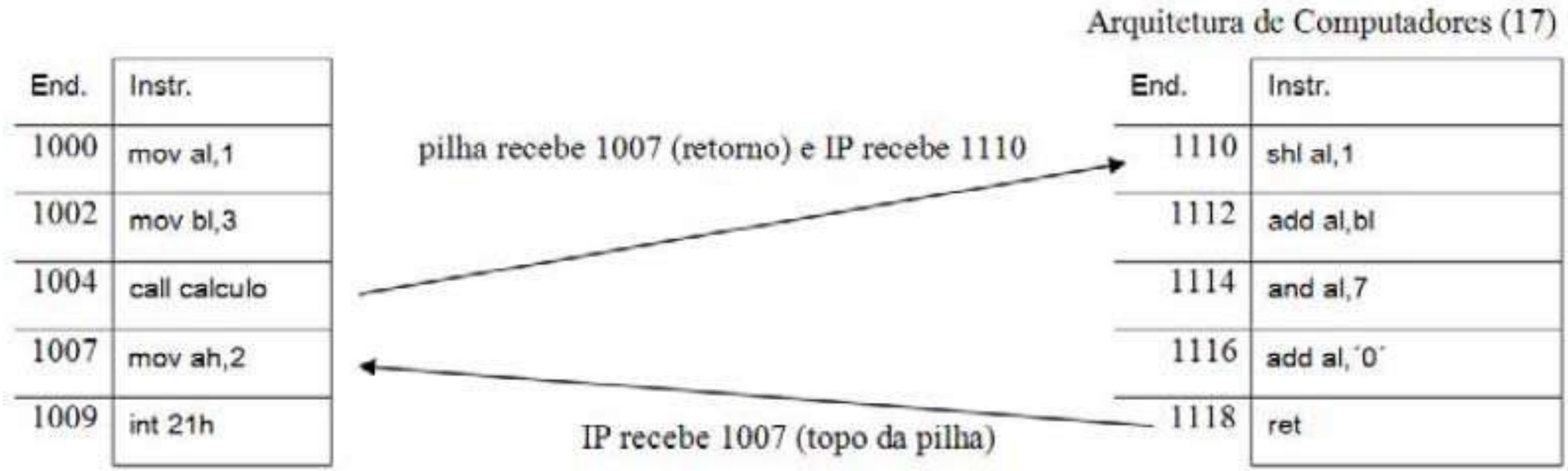
- Os saltos estão associados ao desvio do fluxo de programa de um determinado ponto para outro.
- **Instruções: JMP (*jump*)** - GOTO

Instruções de chamadas de sub-rotinas

- Instrução **CALL:** chama uma sub-rotina, alterando o fluxo normal de execução.
- Endereço de retorno é colocado na pilha pela instrução (**conteúdo de PC é armazenado na pilha - empilhado**).
- Sintaxe: **CALL Proc**

- Instrução **RET**: encerra uma sub-rotina, retomando a execução do programa chamador da sub-rotina.
- Transfere o fluxo de processamento para a instrução seguinte à chamada da sub-rotina (Desempilha o endereço armazenado na pilha e o coloca no registrador PC).
- Sintaxe: **RET**

Fluxo de chamadas de sub-rotinas



Procedimentos

- São estruturas que **agrupam** um conjunto de comandos, que são executados quando o procedimento **é chamado**.
- **Exemplo:**

```
.CODE
nome PROC
;
;corpo da procedure -> instruções
;
nome ENDP
;
;outras procedures seguem abaixo, se existirem
```

Procedimentos

❑**NEAR** – procedimento próximo.

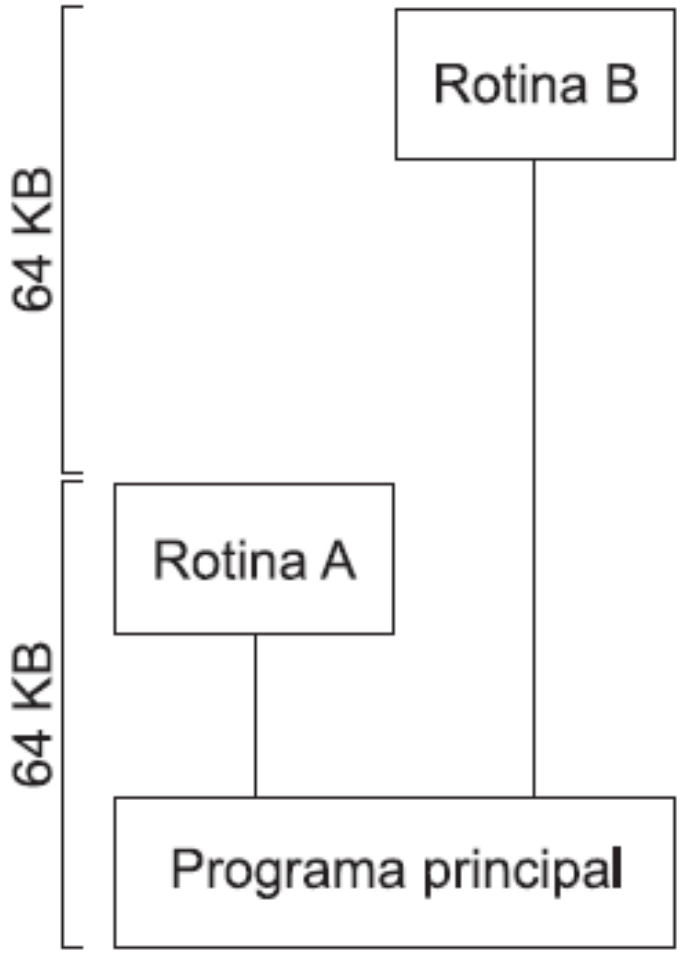
- É utilizado quando o acesso de um procedimento na memória ocorre no mesmo segmento onde se encontra o código de programa em execução (chamada direta **intrassegmento**).
- Alteração do **IP** e sem alteração do **CS**.
- Ao término de um procedimento **NEAR**, o retorno à parte chamadora da sub-rotina é efetuado com a instrução **RET**.

Procedimentos

❑**FAR** – procedimento distante.

- Deve obrigatoriamente ser declarado.
- É utilizado quando o acesso a um procedimento na memória ocorre em um endereço de segmento diferente do endereço de segmento em que se encontra o código do programa em execução (**chamada direta intersegmento**).
- Ocorre alteração do IP e CS.
- Ao término de um procedimento FAR, o retorno à parte chamadora da sub-rotina é efetuado com a instrução **RETF.**

Procedimentos



Os microprocessadores padrão 8086/8088 operam com base em uma arquitetura de memória segmentada, em que somente é possível acessar um segmento de memória de **64 kB** por vez

OBS: - O parâmetro NEAR de procedimentos é utilizado em programas do tipo .COM

-O parâmetro FAR é utilizado em programas do tipo .EXE

Instruções de Salto Condicional

- O salto condicional é diferente de um salto incondicional representado pela instrução **JMP**, pois para ser executado, necessita de uma **instrução condicional** sobre a qual uma decisão será tomada. Normalmente uma condição é estabelecida com o uso de uma das instruções: **CMP** (compare), **AND** (and logical), **OR** (logical or), **NOT** (logical and) e **XOR** (exclusive or).

Instruções de Salto Condicional

Instrução	Significado	Descrição
JA	jump on above	salte se acima de
JAE	jump on above or equal	salte se acima ou igual a
JB	jump on below	salte se abaixo de
JBE	jump on below or equal	salte se abaixo ou igual a
JC	jump on carry	salte se *flag carry* igual a 1
JCXZ	jump if cx register zero	salte se *CX* registra zero
JE	jump on equal	salte se igual a

Instruções de Salto Condicional

Jxxx rótulo_de_destino

xxx define uma condição dependente de algum dos Flags de Estado

- Se a condição xxx é verdadeira:

- a próxima instrução a ser executada é aquela definida pelo rótulo_de_destino;

- a CPU ajusta o registrador IP para apontar para a posição de memória dada por rótulo_de_destino.

- Se a condição xxx é falsa:

- a próxima instrução é aquela que imediatamente segue o salto

Instrução de Comparação

- **CMP** destino, fonte
- **CMP (Compare)** compara os conteúdos destino e fonte, que podem ser:

· registrador e registrador

· registrador e uma posição de memória

· um número diretamente como operando fonte

- Combinações legais de operandos:

	Operando destino	
Operando fonte	Registrador de dados	Posição de memória
Reg. de dados	sim	sim
Posição de memória	sim	**não**
Constante	sim	sim

Instrução de Comparação

❑**Exemplos de instruções válidas:**

- **CMP DX,BX** ; compara os conteúdos de DX e BX
- **CMP AX,[WORD1]** ;compara o conteúdo do registrador AX com o da ;posição de memória WORD1
- **CMP AH,'A'** ;compara o conteúdo de AH com o caracter ASCII 'A'

  ;hexa 41H

Desvio Condicional Simples

- A decisão simples ocorre quando uma ação necessita ser realizada caso uma determinada condição seja **verdadeira**. Se a condição for **falsa**, não deve ser executada a ação especificada como verdadeira, passando o controle operacional do programa para outro ponto, que normalmente também será executado após a ação considerada verdadeira.

Desvio condicional composto

- A decisão composta ocorre quando uma ação precisa ser realizada caso uma determinada condição seja **verdadeira ou falsa**. Se a condição **for verdadeira**, será **executada** uma ação especificada. Se a condição por **falsa**, será executada **outra ação**.

Instruções de Saltos

Tabela 8.1 - Instruções de salto condicional

Instrução	Significado	Descrição
JA	jump on above	salte se acima de
JAE	jump on above or equal	salte se acima ou igual a
JB	jump on below	salte se abaixo de
JBE	jump on below or equal	salte se abaixo ou igual a
JC	jump on carry	salte se *flag carry* igual a 1
JCXZ	jump if cx register zero	salte se *CX* registra zero
JE	jump on equal	salte se igual a

Instruções de Saltos

Instrução	Significado	Descrição
JG	jump on greater	salte se maior que
JGE	jump on greater or equal	salte se maior ou igual a
JL	jump on less	salte se menor que
JLE	jump on less or equal	salte se menor ou igual a
JNA	jump on not above	salte se não acima de
JNAE	jump on not above or equal	salte se não acima ou igual a
JNB	jump on not below	salte se não abaixo de
JNBE	jump on not below or equal	salte se não abaixo ou igual a
JNC	jump on not carry	salte se *flag carry* igual a 0
JNE	jump on not equal	salte se não igual a
JNG	jump on not greater	salte se não maior que
JNGE	jump on not greater or equal	salte se não maior ou igual a
JNL	jump on not less	salte se não menor que
JNLE	jump on not less or equal	salte se não menor ou igual a
JNO	jump or not overlay	salte se não ocorreu *overlay*
JNP	jump on not parity	salte se não for par
JNS	jump on not sign	salte se positivo
JNZ	jump on not zero	salte se não for zero
JO	jump on overflow	salte se ocorreu *overflow*
JP	jump on parity	salte se for par
JPE	jump on parity equal	salte se for par
JPO	jump on parity odd	salte se for ímpar
JS	jump on sign	salte se for negativo
JZ	jump on zero	salte se for zero

Instruções Lógicas

- São elementos importantes quando é preciso trabalhar com mais de uma condição (quando do uso das instruções **AND**, **OR** e **XOR**) ou fazer a negação de uma condição (**instrução NOT**) para a tomada de outra condição.
- De forma operacional as instruções lógicas devem ser usadas seguindo esta sintaxe:

- **AND [destino],[origem]**
- **OR [destino],[origem]**
- **XOR [destino],[origem]**
- **NOT [destino]**

Instruções Lógicas

Tabela 8.7 - Operador AND

Entrada (*bits*)		Saída (*bit*)
Origem	Destino	Destino
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 8.8 - Operador OR

Entrada (*bits*)		Saída (*bit*)
Origem	Destino	Destino
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 8.9 - Operador XOR

Entrada (*bits*)		Saída (*bit*)
Origem	Destino	Destino
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 8.10 - Operador NOT

Entrada (*bits*)	Saída (*bit*)
Destino	Destino
0	1
1	0

Instruções Lógicas

Tabela 8.11 - Comparação entre operadores lógicos

AL	BL	AND AL, BL	OR AI, BL	XOR AL, BL	NOT AL
		AL	AL	AL	AL
1111 1111	1111 1111	1111 1111	1111 1111	0000 0000	0000 0000
1111 0000	0000 1111	0000 0000	1111 1111	1111 1111	0000 1111
1100 0011	1010 1010	0000 0010	1110 1011	0110 1001	0011 1100
0110 0011	0000 0011	0000 0011	0110 0011	0110 0000	1001 1100

OBS: Um detalhe importante a ser considerado é que as instruções lógicas **AND**, **OR**, **XOR** e **NOT** devem ser utilizadas de forma semelhante à instrução **CMP**. Normalmente, após a definição de uma dessas instruções há necessidade de utilizar uma instrução de desvio condicional.

Repetições

- Outra estrutura de programação muito utilizada e importante consiste nos laços de repetição (**loopings ou malhas de repetição**), cuja característica operacional **é executar um trecho de programa por determinado número de vezes**.

Tabela 8.13 - Instruções para execução de laços

Instrução	Significado	Descrição
LOOP	Loop	Laço iterativo
LOOPE	Loop while equal	Enquanto laço igual a
LOOPNE	Loop while not equal	Enquanto laço não for igual a
LOOPNZ	Loop while not zero	Enquanto laço não for zero
LOOPZ	Loop while zero	Enquanto laço for zero

Repetições

- O laço de repetição baseado na instrução **LOOP** é do **tipo iterativo**, ou seja, executa a ação do laço de repetição um determinado número de vezes, semelhante ao laço de repetição **FOR** encontrado em linguagens de alto nível.
- Os laços de repetição **LOOPE, LOOPZ, LOOPNE e LOOPNZ** são do tipo **interativo**, ou seja, são laços condicionais, pois para operarem dependem do valor sinalizado no registrador de estado ZF. Essa forma é semelhante ao laço de repetição **WHILE** encontrado em linguagens de alto nível.
- **OBS:** É bom relembrar que as instruções de laço (sejam elas quais forem) usam o valor que estiver armazenado no registrador geral CX para a operação do contador de passos. O valor é sempre decrementado do registrador geral CX.

Repetições

```
unsigned int count = 5;
unsigned int x=10;
main()
{
   do{
        x++;
        count--;
   }while (count>0)

}
```

```
.MODEL SMALL
.STACK
.CODE
        MOV CX,count    ; CX <- count
L1:     INC x           ; x++
        LOOP  L1        ; repete a partir de L1

.DATA
x       dw      10
count   dw      5

END
```