Linguagens de Montagem

DEMAC – Departamento de Estatística
Matemática Aplicada e Computação
UNESP – Rio Claro

Prof. Daniel Carlos Guimarães Pedronette

Aula 4.

Instruções de Desvio

Desvio Incondicional

- Instrução **JMP**
- Indica ao processador o label da próxima instrução a ser executada

```
jmp     label
```

Instrução JMP

- O que ocorre nesse trecho?

```
        mov     EAX,1
inc_again:
        inc     EAX
        jmp     inc_again
        mov     EBX,EAX
            . . .
```

Desvio Condicional

- Desvia para a instrução identificada por *label* apenas quando a condição especificada for atendida
  - Geralmente a condição testada é o resultado da última operação lógico-aritmética

```
j<cond>    label
```

Desvio Condicional

- Exemplo:

```
read_char:
      mov    DL,0
        . . .
      (code for reading a character into AL)
        . . .
      cmp    AL,0DH       ;compare the character to CR
      je     CR_received  ;if equal, jump to CR_received
      inc    CL           ;otherwise, increment CL and
      jmp    read_char    ;go back to read another
                          ; character from keyboard
CR_received:
      mov    DL,AL
        . . .
```

Desvio Condicional

- Desvio condicional baseado em flags:

Mnemonic	Meaning	Jumps if
Testing for zero:		
jz	jump if zero	ZF = 1
je	jump if equal	
jnz	jump if not zero	ZF = 0
jne	jump if not equal	
jecxz	jump if ECX = 0	ECX = 0 (no flags tested)
Testing for carry:		
jc	jump if carry	CF = 1
jnc	jump if no carry	CF = 0
Testing for overflow:		
jo	jump if overflow	OF = 1
jno	jump if no overflow	OF = 0

Desvio Condicional

- Desvio condicional baseado em flags:

Testing for sign:		
js	jump if (negative) sign	SF = 1
jns	jump if no (negative) sign	SF = 0
Testing for parity:		
jp	jump if parity	PF = 1
jpe	jump if parity is even	
jnp	jump if not parity	PF = 0
jpo	jump if parity is odd	

Desvio Condicional

- Ao realizar uma comparação é necessário saber se os números são com ou sem sinal:

AL = 10110111B and DL = 01101110B

```
cmp     AL,DL
```

AL > DL	AL = 183D DL = 110D	Sem sinal, OK!
AL < DL	−73D +110D	Com sinal, erro!

Desvio Condicional

- Tipo de relacionamentos

$$\begin{aligned}
&\text{num1} = \text{num2} \\
&\text{num1} \neq \text{num2} \\
&\text{num1} > \text{num2} \\
&\text{num1} \geq \text{num2} \\
&\text{num1} < \text{num2} \\
&\text{num1} \leq \text{num2}
\end{aligned}$$

Desvio Condicional

- Instruções de desvio condicional baseadas em comparações sem sinal:

Mnemonic	Meaning	Condition tested
je jz	jump if equal jump if zero	ZF = 1
jne jnz	jump if not equal jump if not zero	ZF = 0
ja jnbe	jump if above jump if not below or equal	CF = 0 and ZF = 0
jae jnb	jump if above or equal jump if not below	CF = 0
jb jnae	jump if below jump if not above or equal	CF = 1
jbe jna	jump if below or equal jump if not above	CF = 1 or ZF = 1

Desvio Condicional

- Instruções de desvio condicional baseadas em comparações com sinal:

Mnemonic	Meaning	Condition tested
je jz	jump if equal jump if zero	ZF = 1
jne jnz	jump if not equal jump if not zero	ZF = 0
jg jnle	jump if greater jump if not less or equal	ZF = 0 and SF = OF
jge jnl	jump if greater or equal jump if not less	SF = OF
jl jnge	jump if less jump if not greater or equal	SF ≠ OF
jle jng	jump if less or equal jump if not greater	ZF = 1 or SF ≠ OF

Desvio Condicional

- Com sinal:

```
CMP AL, BL          ; AL-BL
JG <label>          ; ZF=0 and SF=OF
```

AL	BL	Expressão	Valor	SF	OF	AL > BL
-10	-20	-10-(-20)	+10	SF=0	OF=0	True
-10	-5	-10-(-5)	-5	SF=1	OF=0	False
10	5	10-5	+5	SF=0	OF=0	True
10	20	10-20	-20	SF=1	OF=0	False
10	-2	10-(-2)	+12	SF=0	OF=0	True
100	-90	100-(-90)	+190	SF=1	OF=1	True

Desvio Condicional

- Exemplo:

```
go_back:
        inc     AL
          . . .
          . . .
        cmp     AL,BL
        statement_1
        mov     BL,77H
```

Desvio Condicional

- Exemplo das diferentes instruções:

statement_1	AL	BL	Action taken
`je go_back`	56H	56H	Program control transferred to `inc AL`
`jg go_back`	56H	55H	Program control transferred to `inc AL`
`jg go_back` `jl go_back`	56H	56H	No jump; executes `mov BL,77H`
`jle go_back` `jge go_back`	56H	56H	Program control transferred to `inc AL`

`jne go_back`
`jg go_back`
`jge go_back`
	27H	26H	Program control transferred to `inc AL`

Estruturas de Alto Nível

- Com base nas instruções de desvio condicional, agora podemos implementar estruturas de decisão de linguagens de alto nível:

```
if (condition)
then
        true-alternative
else
        false-alternative
end if
```

Estruturas de Alto Nível

- Condição com operador relacional:

```
if (value1 > value2)
    bigger = value1;
else
    bigger = value2;
```

```
        mov     AX,value1
        cmp     AX,value2
        jle     else_part
then_part:
        mov     AX,value1     ; redundant
        mov     bigger,AX
        jmp     SHORT end_if
else_part:
        mov     AX,value2
        mov     bigger,AX
end_if:
                . . .
```

Estruturas de Alto Nível

- Condição com operador relacional:

```
if (value1 > value2)
    bigger = value1;
else
    bigger = value2;
```

```
        mov     AX,value1
        cmp     AX,value2
        jle     else_part
then_part:
        mov     AX,value1     ; redundant
        mov     bigger,AX
        jmp     SHORT end_if
else_part:
        mov     AX,value2
        mov     bigger,AX
end_if:
                . . .
```

Estruturas de Alto Nível

- Condição com operador AND lógico e relacional:

```
if ((ch >= 'a') && (ch <= 'z'))
    ch = ch - 32;
```

```
        cmp     DL,'a'
        jb      not_lower_case
        cmp     DL,'z'
        ja      not_lower_case
lower_case:
        mov     AL,DL
        add     AL,224
        mov     DL,AL
not_lower_case:
            . . .
```

Estruturas de Alto Nível

- Condição com operador OR lógico e relacional:

```
if ((index < 1) || (index > 100))
    index = 0;
```

```
cmp     CX,1
jl      zero_index
cmp     CX,100
jle     end_if
zero_index:
        xor     CX,CX       ; CX = 0
end_if:
            . . .
```

Estruturas de Alto Nível

- Estruturas de Repetição: While

```
while(total < 700)
  {
    <loop body>
  }
```

```
        jmp     while_cond
while_body:
            . . .
        < instructions for
          while loop body >
            . . .
while_cond:
        cmp     BX,700
        jl      while_body
end_while:
            . . .
```

Estruturas de Alto Nível

- Estruturas de Repetição: Do While (Repeat)

```
do
  {
    <loop body>
  }
while (number > 0);
```

```
loop_body:
          . . .
      < instructions for
        do-while loop body >
          . . .
cond_test:
      or     DI,DI
      jg     loop_body
end_do_while:
          . . .
```

Estruturas de Alto Nível

- Estruturas de Repetição: For

```
for (i = 0; i < SIZE; i++)    /* for (i = 0 to SIZE-1) */
  {
    <loop body>
  };
```

```
        xor     SI,SI
        jmp     SHORT for_cond
loop_body:
            . . .
        < instructions for
          the loop body >
            . . .
        inc     SI
for_cond:
        cmp     SI,SIZE
        jl      loop_body
            . . .
```

Exemplos

Prática:

- Codificar,
- Montar,
- Linkar e
- Testar!

Exercícios