CS2422 Assembly Language & System Programming

October 17, 2006

Today's Topics

- Conditional Processing
  - If…then…else
  - While…do; Repeat…until

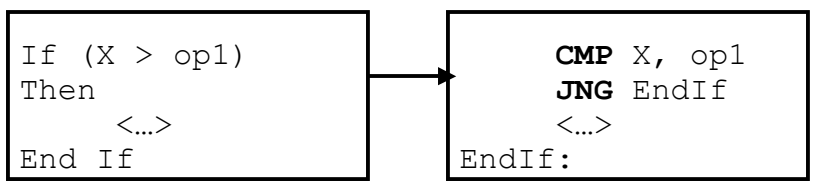
Study Guide

- Suggested Order:
  - Section 6.1: Introduction
  - Section 6.3.1-6.3.4: Conditional Jumps
  - Section 6.5: Conditional Structures
  - Section 6.2: Boolean Operations

  (Optional): Sections 6.3.5, 6.4, 6.6, 6.7

CMP and Jcond Instruction

- The IF statement in C and PASCAL is converted into CMP and J*cond* instructions in x86 Assembly:



CMP Instruction (1 of 3)

- Compares the destination operand to the source operand
  - Nondestructive subtraction of source from destination (destination operand is not changed)
- Syntax: CMP *destination, source*
- Example: destination == source

```
mov al,5
cmp al,5          ; Zero flag set
```

CMP Instruction (2 of 3)

- Example: destination > source

(both the Zero and Carry flags are clear)

```
mov al,6
cmp al,5          ; ZF = 0, CF = 0
```

The comparisons shown so far were unsigned.

CMP Instruction (3 of 3)

The comparisons shown here are performed with signed integers.

- Example: destination > source

```
mov al,5
cmp al,-2 ; Sign flag = Overflow flag
```

- Example: destination < source

```
mov al,-1
cmp al,5 ; Sign flag != Overflow flag
```

J*cond* Instruction

- A conditional jump instruction branches to a label when specific register or flag conditions are met

Examples:

- JB, JC jump to a label if the Carry flag is set
- JE, JZ jump to a label if the Zero flag is set
- JS jumps to a label if the Sign flag is set
- JNE, JNZ jump to a label if the Zero flag is clear
- JECXZ jumps to a label if ECX equals 0

Jumps Based on Specific Flags

Mnemonic	Description	Flags
JZ	Jump if zero	ZF = 1
JNZ	Jump if not zero	ZF = 0
JC	Jump if carry	CF = 1
JNC	Jump if not carry	CF = 0
JO	Jump if overflow	OF = 1
JNO	Jump if not overflow	OF = 0
JS	Jump if signed	SF = 1
JNS	Jump if not signed	SF = 0
JP	Jump if parity (even)	PF = 1
JNP	Jump if not parity (odd)	PF = 0

Jumps Based on Equality

Mnemonic	Description
JE	Jump if equal (*leftOp* = *rightOp*)
JNE	Jump if not equal (*leftOp* ≠ *rightOp*)
JCXZ	Jump if CX = 0
JECXZ	Jump if ECX = 0

Jumps Based on Unsigned Comparisons

Mnemonic	Description
JA	Jump if above (if *leftOp* > *rightOp*)
JNBE	Jump if not below or equal (same as JA)
JAE	Jump if above or equal (if *leftOp* >= *rightOp*)
JNB	Jump if not below (same as JAE)
JB	Jump if below (if *leftOp* < *rightOp*)
JNAE	Jump if not above or equal (same as JB)
JBE	Jump if below or equal (if *leftOp* <= *rightOp*)
JNA	Jump if not above (same as JBE)

Jumps Based on Signed Comparisons

Mnemonic	Description
JG	Jump if greater (if *leftOp* > *rightOp*)
JNLE	Jump if not less than or equal (same as JG)
JGE	Jump if greater than or equal (if *leftOp* >= *rightOp*)
JNL	Jump if not less (same as JGE)
JL	Jump if less (if *leftOp* < *rightOp*)
JNGE	Jump if not greater than or equal (same as JL)
JLE	Jump if less than or equal (if *leftOp* <= *rightOp*)
JNG	Jump if not greater (same as JLE)

More Frequently Used Jcond Instructions

- JE (Equal)
- JNE (Not Equal)
- JG or JGE (Greater Than or Equal)
- JL or JLE (Less Than or Equal)
- Note: JG=JNLE, JGE=JNL, …etc.

Simple IF

- If (op1=op2) then <…> end if
- Two different approaches:

```
     CMP op1, op2
     JE True
     JMP EndIf
True:
     <…>
EndIf
```

```
     CMP op1, op2
     JNE False
     <…>
False:
```

IF … AND …

```
If    (X > op1)and
      (Y <=op2)and
      …
Then
      <…>
End If
```

→

```
      CMP X, op1
      JNG EndIf
      CMP Y, op2
      JNLE EndIf
      CMP …
      …
      …

      <…>
EndIf:
```

IF … OR …

```
If    (X > op1) or
      (Y <=op2) or
      …
Then
      <…>
End If
```

→

```
      CMP X, op1
      JG True
      CMP Y, op2
      JLE True
      CMP …
      …
      …
      JMP EndIf
True:
      <…>
EndIf:
```

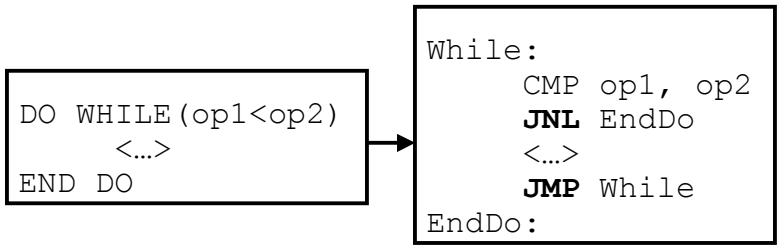
Exercise

- Can you rewrite the last slide and remove the “JMP EndIf”?
- How about adding the “else” part?

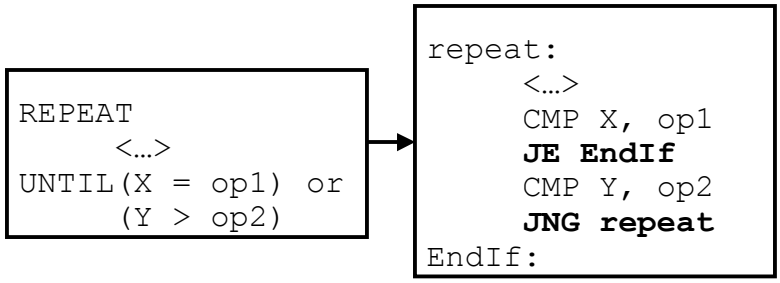
A Strategy for Compound Conditions

- Figure out the “longest path” of the program.
- Then find the “short cuts” (or early jumps).

WHILE

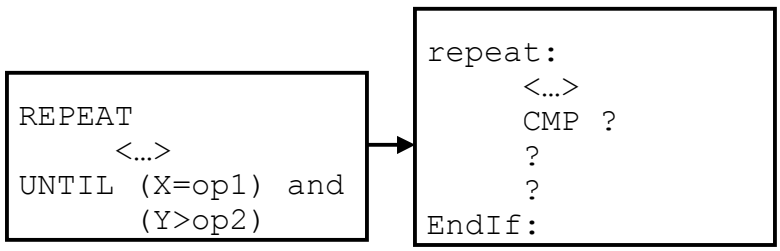


REPEAT UNTIL



More Exercises

- Write the code for the CASE statement.
- AND conditions in REPEAT-UNTIL?



Flags and Jcond

How do Jcond instructions decide which way to go?

- They check the flags!
- Examples:
  - JE/JNE checks Zero flag.
  - JG/JL checks Sign flag.
- CMP instruction sets the flags.

Boolean and Comparison Instructions

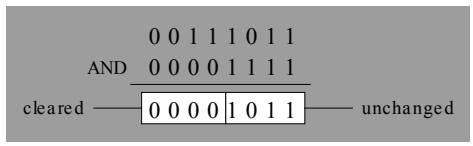
- AND Instruction
- OR Instruction
- XOR Instruction
- NOT Instruction
- TEST Instruction
- CMP Instruction

AND Instruction

- Performs a Boolean AND operation between each pair of matching bits in two operands
- Syntax:

  AND *destination, source*

  (same operand types as MOV)



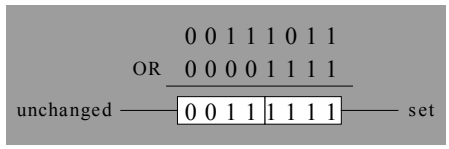
AND

x	y	x ∧ y
0	0	0
0	1	0
1	0	0
1	1	1

OR Instruction

- Performs a Boolean OR operation between each pair of matching bits in two operands
- Syntax:

  OR *destination, source*



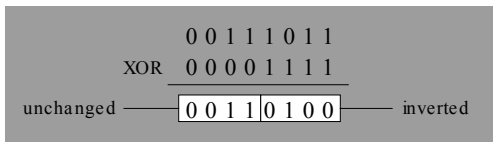
OR

x	y	x ∨ y
0	0	0
0	1	1
1	0	1
1	1	1

XOR Instruction

- Performs a Boolean exclusive-OR operation between each pair of matching bits in two operands
- Syntax:

  XOR *destination, source*



XOR

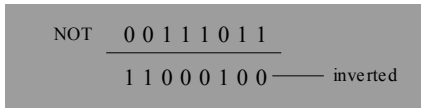
x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

XOR is a useful way to toggle (invert) the bits in an operand.

NOT Instruction

- Performs a Boolean NOT operation on a single destination operand
- Syntax:

NOT *destination*



NOT

x	¬x
F	T
T	F

Application – An Example

- Task: Convert the character in AL to upper case.
- Solution: Use the AND instruction to clear bit 5.

```
mov al,'a`          ; AL = 01100001b
and al,11011111b    ; AL = 01000001b
```

TEST Instruction

- Performs a nondestructive AND operation between each pair of matching bits in two operands
- No operands are modified, but the Zero flag is affected.
- Example: jump to a label if either bit 0 or bit 1 in AL is set.

```
test al,00000011b
jnz  ValueFound
```