Lecture 8: Control Structures

- CMP Instruction
- Conditional Jumps
- High Level Logic Structures

Comparing Values

- The CMP instruction performs a comparison between two numbers using an implied subtraction. This means that the flags (in the flags register) are set to show the result of a subtraction but the numbers subtracted do not change.

Example

- CMP DX, BX ; compare from HW2

  When BX = 0004 and DX = 0008,
  DX – BX = 0004 (remember – implied)
  NV - no overflow
  PL – positive

  When BX = 000A and DX = 0008,
  DX – BX = FFFE (-2)
  NV - no overflow
  NG – negative

Flags Set by CMP

- tables from 6.1.10 in Irvine
**CMP Examples**

- AX = 10, BX = -12 (decimal)
- CMP AX, BX
  AX – BX = +22
  PL (positive), CY (carry), NV (no overflow), NZ (not zero)
- CMP BX, AX
  BX – AX = -22
  NG (negative), NC (no carry), NV (no overflow), NZ (not zero)
- CMP AX, AX
  AX – AX = 0
  PL (positive), NC (no carry), NV (no overflow), ZR (zero)

**What can we compare?**

- register to register:
  – CMP AX, BX
- register to memory:
  – CMP AX, mval
- register to immediate:
  – CMP AX, 42
- memory to register:
  – CMP mval, AX
- memory to immediate (!)
  – CMP mval, 42

**What can’t we compare?**

- You can not compare memory to memory!!!
- One value will need to be copied into a register prior to the CMP instruction.

**Why is this Useful?**

- CMP is generally followed by a conditional jump statement to create an If statement:
  CMP dest, src :sets flags
  Jxxx label :jumps based on flags
Conditional Jumps

• Conditional jumps are used to jump to another location based on the settings in the flags register.
• The numbers you are comparing can represent signed or unsigned values. Different flags will be checked depending on which interpretation you are using.
• How does the CPU know how you are interpreting the numbers?
  – It knows by your choice of jump instruction!

General Comparison Jumps

• Irvine, Ch 6, table 4
  • These are the same for signed and unsigned

Unsigned Comparison Jumps

• Irvine, Ch 6, table 5
  • Unsigned jumps refer to “above and “below”

Signed Comparison Jumps

• Irvine, Ch 6, Table 6
  • Signed jumps refer to “greater” and “less”
Signed vs. Unsigned

.data
total dw 0FFFFh
; jump if total < 10 (signed)
CMP total, 10
  JL less10 ; jump total < 10
  ...
less10:

FFFFh = -1, so code will jump to less10 since −1 < 10.

; jump if total < 10 (unsigned)
CMP total, 10
  JB less10 ; jump total < 10
  ...
less10:

This code will not jump to less10 because FFFFh unsigned = 65535 > 10.

How does assembler know if FFFFh is −1 or 65,535? You tell it by your choice of jump instruction!

Using Conditional Jumps

• As shown earlier, the relation expressed by the jump instruction refers to the two operands from a previous CMP.
• Conditional jumps are usually used directly after a CMP.
• Why usually? Well, you could use a jump based on the result of an arithmetic operation.

Example

CMP DX, BX ; compare from HW2
JGE add_lup ; jump to top of loop

When BX = 0004 and DX = 0008,
  DX – BX = 0004 (remember – implied)
  NV - no overflow (0)
  PL – positive (0)
  overflow matches sign – jumps back to
top of loop: DX >= BX

When BX = 000A and DX = 0008,
  DX – BX = FFFE (-2)
  NV - no overflow (0)
  NG – negative (1)
  overflow <> sign – does not jump:
  DX < BX

High Level Logic Structures

• So what are some of the control structures in high level programming languages?
  – if
  – do-while
  – repeat-until
  – case
  – ….
• These can be implemented in assembly using CMP and conditional Jump
If Statement

if (op1 = op2) then
  <statement1>
  <statement2>
end if

In assembler (still pseudo-code!):
  cmp op1, op2
  jne false
  <statement1>
  <statement2>
false: <rest of program>

If Statement Example

.data
op1 db 10
op2 db −12
op3 db ?

.code
  mov al, op1  ; why?
  cmp al, op2  ; op1 = op2?
  jne noteq  ; if no, jump
  mov bl, op2  ; statement 1
  mov op3, bl  ; statement 2
noteq: add al, op2

If-then-Else

if (temp > max) then
  max = temp
else
  max = max + 1
endif

In Assembly:
  mov ax, temp
  mov bx, max
  cmp ax, bx ; compare temp to max
  ; "if"
  jle els  ; jump if temp <= max
  mov max, ax ; temp > max "then"
  jmp done ; unconditional jump
els: inc bx  ; temp <= max "else"
  mov max, bx
done:

Compound If Using OR

• Examples from Irvine, 6.4.2
Compound IF Using AND

- more examples from Irvine 6.4.2

Another example (this time: unsigned)

if ((ax < 10) and (bx < 10)) then
  assign 1 to CX register
else
  assign 0 to CX register
end if

In assembly:

```
cmp ax, 10
jae els ; jump ax >= 10
cmp bx, 10 ; ax < 10
jae els ; jump bx >= 10
mov cx, 1 ; ax < 10 and bx < 10
jmp done
els: mov cx, 0 ; ax >= 10 or bx >=10
done:
```

With AND – negate the conditions you test for!

Do-While

do
  ax = ax + 1
  cx = ax
while ((ax < bx) AND (cx == dx))

In assembly:

```
top: inc ax ; ax = ax + 1
mov cx, ax ; cx = ax
cmp ax, bx
jae done ; ax >= bx done
mov cx, dx
jne done ; cx <> dx done
jmp top
done:
```

The condition that brings you back to the top is (AX < BX) AND (CX == DX).
You want to exit from the loop when AX >= BX or CX <> DX

Case Statement

Examples in Irvine, 6.4.5