AACS3064
Computer Systems Architecture

Chapter 9: Conditional Processing

Chapter Overview

1) Boolean and Comparison instruction
   - CMP
   - AND, OR, XOR, NOT, TES

2) Conditional Jumps
   - Conditional Jumps : Jcond(JB, JC, JE, JZ, etc)

3) Conditional Loop
   - Conditional Loop : LOOPZ, LOOPE, LOOPNZ, LOOPNE

1. Boolean and Comparison instruction

1. Boolean and Comparison instruction

AND **instruction**

- Boolean AND operation

  For each matching bit in the two numbers:

  - If both bits are 1, the result bit is 1.
  - Otherwise, the result bit is 0.

- E.g.

```
X:      1111 1111
Y:      0001 1100
-------------------
X^Y     0001 1100
```

X	Y	X^Y
0	0	0
0	1	0
1	0	0
1	1	1

AND truth table

1. Boolean and Comparison instruction (Continued)

AND **instruction**

- Performs a Boolean AND operation between 2 operands and places the result in the destination operand.
- Format :

```
AND  destination, source
```

- E.g.

```
MOV  AL, 00001100B
AND  AL, 11110100B ; AL = ?
```

1. Boolean and Comparison instruction (Continued)

OR instruction

- Boolean OR operation

  For each matching bit in the two numbers:

  - If one of the input bits is 1, the result bit is 1.
  - Otherwise, the result bit is 0.

- E.g.

```
X:        1110 1100
Y:        0001 1100
-------------------
X ∨ Y     1111 1100
```

X	Y	X ∨ Y
0	0	0
0	1	1
1	0	1
1	1	1

OR truth table

1. Boolean and Comparison instruction (Continued)

OR instruction

- Performs a Boolean OR operation between 2 operands and places the result in the destination operand.
- Format :

```
OR   destination, source
```

- E.g.

```
MOV  AL, 01011100B
OR   AL, 00000100B ; AL = ?
```

1. Boolean and Comparison instruction (Continued)

XOR instruction

- Boolean XOR operation

  For each matching bit in the two numbers:

  - If both bits are the same, the result bit is 0.
  - Otherwise, the result bit is 1.

- E.g.

```
X:        1011 0111
Y:        0001 1100
-------------------
X ⊕ Y     1010 1011
```

X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0

XOR truth table

1. Boolean and Comparison instruction (Continued)

XOR instruction

- Performs a Boolean XOR operation between 2 operands and places the result in the destination operand.
- Format :

```
XOR  destination, source
```

- E.g.

```
MOV  AL, 11011000B
XOR  AL, 01001100B ; AL = ?
```

1. Boolean and Comparison instruction (Continued)

NOT **instruction**

- Boolean NOT operation

  toggles (inverts) all bits.

- E.g.

```
  X:      1001 1101
-------------------
 ~X:      0110 0010
```

X	~X
0	1
1	0

NOT truth table

1. Boolean and Comparison instruction (Continued)

NOT **instruction**

- Performs a Boolean NOT operation in an operand.
- The result is **one's complement**.
- Format :

```
NOT  register / memory
```

- E.g.

```
MOV  AL, 11110000B
NOT  AL                ; AL = ?
```

1. Boolean and Comparison instruction (Continued)

TEST **instruction**

- Sets the flags like AND operation but does not modify the destination operand.
- Zero flag is affected.

- Format :

```
TEST destination , source
```

- E.g.

```
MOV  AL, 11110000B
TEST AL, 00001001B        ; AL = ?
```

1. Boolean and Comparison instruction (Continued)

TEST **instruction**

- E.g.

```
TEST BL, 11110000B ; Any of the leftmost
JNZ  ...           ; bits in BL nonzero?
```

```
TEST AL, 00000001B ; Does the Al contain
JNZ  ...           ; an odd number?
```

```
TEST DX, 0FFH      ; Does the DX contain
JZ   ...           ; a zero value?
```

1. Boolean and Comparison instruction (Continued)

CMP **instruction**

- Used to compare two numeric data values.
- Performs an implied subtraction of a source operand from a destination operand.
- Affects the AF, CF, OF, PF, SF and ZF flags.
- Format :

```
CMP        destination, source
```

1. Boolean and Comparison instruction (Continued)

CMP **instruction**

- Compare between two unsigned operands :

CMP Results	ZF	CF
Destination < source	0	1
Destination > source	0	0
Destination = source	1	0

- Compare between two signed operands :

CMP Results	Flags
Destination < source	SF ≠ OF
Destination > source	SF = OF
Destination = source	ZF = 1

1. Boolean and Comparison instruction (Continued)

CMP **instruction**

- E.g. Subtracting 10 from 5 requires a borrow.

```
MOV  AX, 5
CMP  AX, 10     ; ZF = 0 and CF = 1
```

- E.g. Subtracting the source from the destination produces zero :

```
MOV  AX, 1000
MOV  CX, 1000
CMP  CX, AX     ; ZF = 1 and CF = 0
```

1. Boolean and Comparison instruction (Continued)

CMP **instruction**

- E.g. Subtracting 0 from 105 generate a positive, nonzero value.

```
MOV  SI, 105
CMP  SI, 0              ; ZF = 0 and CF = 0
```

2. Conditional Jumps

2. Conditional Jumps

Conditional Structures

- Logic structure can be implemented using a combination of comparisons and jumps.

- 2 Steps :
  1. An operation such as CMP or AND modifies the CPU status flags.
  2. A Conditional jump instruction tests the flags and causes a branch to a new address.

2. Conditional Jumps (Continued)

Conditional Structures

- E.g.

JZ instruction jumps to label **L1** if the ZF was set.

```
        CMP  AX, 0
        JZ   L1    ; jump if ZF = 1
        :
        :
L1 :
```

2. Conditional Jumps (Continued)

Conditional Structures

- E.g.

JNZ instruction jumps to label **L2** if the ZF is clear.

```
        AND  DL, 10110000B
        JNZ  L2                ; jump if ZF = 0
             :
             :
L2 :
```

2. Conditional Jumps (Continued)

Jcond Instruction

- A conditional jump instruction branches to a destination label when a status flag condition is true.
- Otherwise, the instruction immediately following the conditional jump is executed.
- Format :

Jcond	**destination**

2. Conditional Jumps (Continued)

Types of Conditional jump Instructions

- Four groups :
  - Jumps based on specific flag values
  - Jumps based on equality between operands or the value of CX
  - Jumps based on comparisons of unsigned operands.
  - Jumps based on comparisons of signed operands.

2. Conditional Jumps (Continued)

Jumps Based on Specific Flag Values

Mnemonic	Description	Flags / Registers
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

2. Conditional Jumps (Continued)

Jumps Based on Equality

Mnemonic	Description
JE	Jump if equal
JNE	Jump if not equal
JCXZ	Jump if CX = 0

2. Conditional Jumps (Continued)

Jumps Based on Equality

- E.g.

```
MOV       DX, 1123H
CMP       DX, 1123H
JNE       L5                ; jump not taken
JE        L1                ; jump is taken
```

- E.g.

```
MOV       CX, FFFFH
INC       CX
JCXZ      L2                ; jump is taken
```

2. Conditional Jumps (Continued)

Jumps Based on Unsigned Comparisons

Mnemonic	Description
JA	Jump if above (if *destination* > *source*)
JNBE	Jump if not below or equal
JAE	Jump if above or equal (if *destination* ≥ *source*)
JNB	Jump if not below
JB	Jump if below (if *destination* < *source*)
JNAE	Jump if not above or equal
JBE	Jump if below or equal (if *destination* ≤ *source*)
JNA	Jump if not above

2. Conditional Jumps (Continued)

Jumps Based on Signed Comparisons

Mnemonic	Description
JG	Jump if greater (if *destination* > *source*)
JNLE	Jump if not less than or equal
JGE	Jump if greater than or equal (if *destination* ≥ *source*)
JNL	Jump if not less
JL	Jump if less (if *destination* < *source*)
JNGE	Jump if not greater than or equal
JLE	Jump if less than or equal (if *destination* ≤ *source*)
JNG	Jump if not greater

2. Conditional Jumps (Continued)

Jumps Based on Unsigned Comparisons

- E.g.

```
MOV       AL, +127
CMP       AL, -128
JA        IsAbove   ; jump not taken, 7FH < 80H
JG        IsGreater; jump taken, +127 > -128
```

2. Conditional Jumps (Continued)

Jumps Based on Signed Comparisons

- E.g.

```
MOV     BX, +32
CMP     BX, -35
JNG     L5              ; jump not taken
JNGE    L5              ; jump not taken
JGE     L1              ; jump is taken
```

- E.g.

```
MOV     CX, 0
CMP     CX, 0
JG      L5              ; jump not taken
JNL     L1              ; jump is taken
```

2. Conditional Jumps (Continued)

Testing Multiple Conditions

- E.g. Determine if all conditions are true (AND)

```
                CMP         AL, BL
                JNE      not_equal
                CMP         AL, BH
                JNE      not_equal
                CMP          AL,CL
                JNE      not_equal

equal :              <processing>
                            .....
not_equal:           <processing>
```

3. Conditional Loop

3. Conditional Loop

LOOPZ and LOOPE **instructions**

- Loop if zero and Loop if equal.
- Continue looping as long as CX is zero or the zero condition is set.
- Useful when scanning an array for the first element that does not match a given value.

3. Conditional Loop (Continued)

LOOPNZ and LOOPNE **instructions**

- Loop if not zero and Loop if not equal.
- Continue looping as long as CX is not zero or the zero condition is not set.
- Useful when scanning an array for the first element that matches a given value.

3. Conditional Loop (Continued)

Application

- Sums the values in an array

```
.DATA
      myData      BYTE  1, 2, 3, 4
      total       BYTE  0
.CODE
      MOV   AX, 0
      MOV   BX, OFFSET myData
      MOV   CX, 03H
A20:
      ADD   AL, [BX]
      INC   BX
      DEC   CX
      JNZ   A20
      MOV   total , AL
```

3. Conditional Loop (Continued)

Application

- Move the first character into DL.

```
.DATA
      myData      BYTE  2, 4, 7, 'H', 'A', 9
.CODE
      ; write your code here
```

Chapter Review

1) Boolean and Comparison instruction
   - CMP
   - AND, OR, XOR, NOT, TES

2) Conditional Jumps
   - Conditional Jumps : Jcond(JB, JC, JE, JZ, etc)

3) Conditional Loop
   - Conditional Loop : LOOPZ, LOOPE, LOOPNZ, LOOPNE