Chapter 6 Program Control Instructions

Assoc. Prof. Dr. Gazi Erkan BOSTANCI

Slides are mainly based on The Intel Microprocessors by Barry B. Brey, 2008

- The program control instructions direct the flow of a program and allow the flow to change.
- A change in flow often occurs after a decision made with the CMP and TEST instruction followed by a conditional jump instruction.

The Jump Group

- The main program control instruction, jump (JMP), allows the programmer to skip sections of a program and branch to any part of memory for next instruction.
- A conditional jump instruction allows the programmer to make decisions based upon numerical tests. The results of numerical tests are held in the flag bits, which are then tested by conditional jump instructions.

Unconditional Jump (JMP)

- Whenever a jump instruction references an address, a label normally identifies the address.

```
        XOR BX, BX
START:  MOV AX,1
        ADD AX, BX
        JMP NEXT
...
NEXT:   MOV BX, AX
        JMP START
```

Conditional Jumps and Conditional Sets

- The conditional jump instructions test the following flag bits: sign (s), zero (z), carry (c), parity (p) and overflow (o).
  - If the condition under test is true, a branch to a label associated with the jump instruction occurs.
  - If the condition is false, the next sequential step in the program executes.

Assembly Language	*Tested Condition*	*Operation*
JA	Z = 0 and C = 0	Jump if above
JAE	C = 0	Jump if above or equal
JB	C = 1	Jump if below
JBE	Z = 1 or C = 1	Jump if below or equal
JC	C = 1	Jump if carry
JE or JZ	Z = 1	Jump if equal or jump if zero
JG	Z = 0 and S = 0	Jump if greater than
JGE	S = 0	Jump if greater than or equal
JL	S != O	Jump if less than
JLE	Z = 1 or S != O	Jump if less than or equal
JNC	C = 0	Jump if no carry
JNE or JNZ	Z = 0	Jump if not equal or jump if not zero
JNO	O = 0	Jump if no overflow
JNS	S = 0	Jump if no sign (positive)
JNP or JPO	P = 0	Jump if no parity or jump if parity odd
JO	O = 1	Jump if overflow
JP or JPE	P = 1	Jump if parity or jump if parity even
JS	S = 1	Jump if sign (negative)
JCXZ	CX = 0	Jump if CX is zero
JECXZ	ECX = 0	Jump if ECX equals zero
JRCXZ	RCX = 0	Jump if RCX equals zero (64-bit mode)

LOOP

- The LOOP instruction is a combination of a decrement CX and JNZ conditional jump.
  - LOOP decrements CX;
    - If CX!=0, it jumps to the address indicated by the label.
    - If CX becomes 0, the next sequential instruction executes.
- The following example shows how data in one block of memory (BLOCK1) add to data in a second block of memory (BLOCK2), using LOOP to control how many numbers add.
  - The LODSW and STOSW instructions access the data in BLOCK1 and BLOCK2.
  - The ADD AX, ES:[DI] instruction accesses the data in BLOCK2 located in the extra segment. The only reason that BLOCK2 is in the extra segment is that DI addresses extra segment data for the STOSW instruction.
- In this example, the extra segment also addresses data in the data segment, so the contents of DS are copied to ES through the accumulator. Unfortunately, there is no direct move from segment register to segment register instruction.

```
        .DATA ;start data segment
        ; DUP duplicates ? 100 times,
        ; ? Means uninitialized
        ; 4 DUP(2) = 2,2,2,2
        BLOCK1 DW 100 DUP(?) ;100 words for BLOCK1
        BLOCK2 DW 100 DUP(?) ;100 words for BLOCK2
        .CODE ;start code segment
        .STARTUP ;start program
        MOV AX,DS ;overlap DS and ES
        MOV ES,AX
        CLD ;select auto-increment
        MOV CX,100 ;load counter
        MOV SI,OFFSET BLOCK1 ;address BLOCK1
        MOV DI,OFFSET BLOCK2 ;address BLOCK2
L1:     LODSW ;load AX with BLOCK1
        ADD AX,ES:[DI] ;add BLOCK2
        STOSW ;save answer
        LOOP L1 ;repeat 100 times
```

Procedures

- The procedure (subroutine, function or method) is an important part of any computer system's architecture.
- A procedure is a group of instructions that usually performs one task.
- It is a reusable section of software that is stored in memory once, but used often as necessary.
- The CALL instruction links to the procedure, and the RET (return) instruction returns from the procedure.
- The stack stores the return address whenever a procedure is called during the execution of a program.
  - The CALL instruction pushes the address of the instruction following the CALL (return address) on the stack.
  - The RET instruction removes an address from the stack so the program returns to the instruction following the CALL.

- Ex.

```
SUMS  PROC
      ADD AX, BX
      ADD AX, CX
      ADD AX, DX
      RET
SUMS  ENDP
```

Procedure Definition

```
CALL  SUMS
```

Procedure Call

Introduction to Interrupts

- An interrupt is either a hardware-generated CALL (externally derived from a hardware signal) or a software-generated CALL (internally derived from the execution of an instruction or aby some other internal event.)
- At time, an interrupt is called an exception. Either type interrupts the program by calling an interrupt service procedure (ISP) or interrupt handler.
  - Interrupts are special types of CALL instructions.

Interrupt Vectors

- An **interrupt vector** is a 4-byte number stored in the first 1024 bytes of the memory (00000H–003FFH) when the microprocessor operates in the real mode.

Number	*Address*	*Microprocessor*	*Function*
0	0H–3H	All	Divide error
1	4H–7H	All	Single-step
2	8–BH	All	NMI pin
3	CH–FH	All	Breakpoint
4	10H–13H	All	Interrupt on overflow
5	14H–17H	80186–Core2	Bound instruction
6	18H–1BH	80186–Core2	Invalid opcode
7	1CH–1FH	80186–Core2	Coprocessor emulation
8	20H–23H	80386–Core2	Double fault
9	24H–27H	80386	Coprocessor segment overrun
A	28H–2BH	80386–Core2	Invalid task state segment
B	2CH–2FH	80386–Core2	Segment not present
C	30H–33H	80386–Core2	Stack fault
D	34H–37H	80386–Core2	General protection fault (GPF)
E	38H–3BH	80386–Core2	Page fault
F	3CH–3FH	—	Reserved
10	40H–43H	80286–Core2	Floating-point error
11	44H–47H	80486SX	Alignment check interrupt
12	48H–4BH	Pentium–Core2	Machine check exception
13–1F	4CH–7FH	—	Reserved
20–FF	80H–3FFH	—	User interrupts

- Samples
  - Loop
  - Call-Ret
  - Absolute Difference
  - Reverse
  - Sort