LECTURE # 06

PREPARED BY: TAHMINA KHAN

MORE SAMPLE PROGRAMS

Write a program that adds 5 bytes of data and saves the result. The data should be the hex numbers given as: 25, 12, 15, 1F and 2B.

PROGRAM:

```
                        .MODEL SMALL
                        .STACK 64
                        .DATA
DATA_IN     DB 25H,12H, 15H, 1FH, 2BH
SUM                     DB ?
                        .CODE
MAIN                    PROC FAR
                        MOV AX, @DATA
                        MOV DS, AX
                        MOV CX, 05
                        MOV BX, OFFSET DATA_IN
                        MOV AL,0
AGAIN:                  ADD AL, [BX]
                        INC BX
                        DEC CX
                        JNZ AGAIN
                        MOV SUM, AL
                        MOV AH,4CH
                        INT21H
MAIN                    ENDP
                        END MAIN
```

MORE SAMPLE PROGRAMS

Write and run a program that adds four words of data and saves the result. The values will be 234DH, 1DE6H, 3BC7H and 566AH.

PROGRAM:

```
                    .MODEL SMALL
                    .STACK 64
                    .DATA
DATA_IN             DW 234DH, 1DE6H, 3BC7H, 566AH
                    ORG 10H
SUM                 DW ?
                    .CODE
MAIN                PROC FAR
                    MOV AX, @DATA
                    MOV DS,AX
                    MOV CX, 04
                    MOV DI, OFFSET DATA_IN
                    MOV BX,00
ADD_LP:             ADD BX,[DI]
                    INC DI
                    INC DI
                    DEC CX
                    JNZ ADD_LP
                    MOV SI, OFFSET SUM
                    MOV [SI], BX
                    MOV AH, 4CH
                    INT 21H
MAIN                ENDP
                    END MAIN
```

MORE SAMPLE PROGRAMS

Write and run a program that transfers 6 bytes of data from memory locations with offset of 0010H to memory locations with offset of 0028H.

PROGRAM:

```
                    .MODEL SMALL
                    .STACK 64
                    .DATA
                    ORG 10H
DATA_IN             DB 25H, 4FH, 85H, 1FH, 2BH, C4H
                    ORG 28H
COPY                DB 6 DUP(?)
                    .CODE
MAIN                PROC FAR
                    MOV AX, @DATA
                    MOV DS, AX
                    MOV SI, OFFSET DATA_IN
                    MOV DI, OFFSET COPY
                    MOV CX, 06H
MOV_LOOP:           MOV AL, [SI]
                    MOV [DI], AL
                    INC SI
                    INC DI
                    DEC CX
                    JNZ MOV_LOOP
                    MOV AH, 4CH
                    INT 21H
MAIN                ENDP
                    END MAIN
```

CONTROL TRANSFER INSTRUCTIONS

- In the sequence of instructions, it is often necessary to transfer program control to a different location.

– If control is transferred to a memory location within the current code segment, it is NEAR.

- Sometimes called intrasegment. (within segment)
- The IP is updated and CS remains the same.

– If control is transferred outside the current code segment, it is a FAR jump.

- Or intersegment. (between segments)
- Both CS and IP have to be updated to the new values.

CONTROL TRANSFER INSTRUCTIONS

- The control transfer is handled in Assembly language with the help of 'JMP' command.
- JMP instruction has two types:
  - Conditional jumps
  - Unconditional jumps

CONTROL TRANSFER INSTRUCTIONS

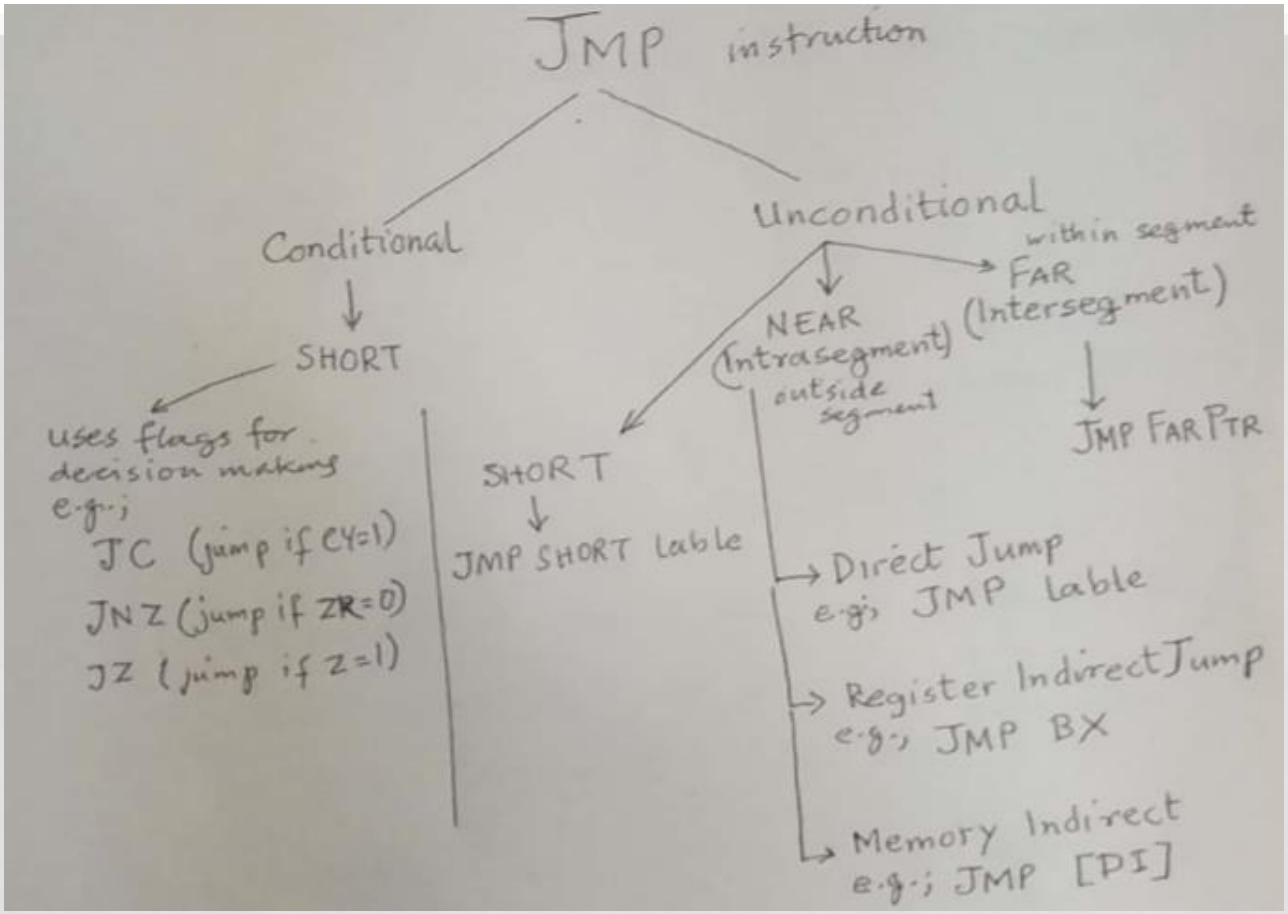

Figure 1: Types of Jumps

CONTROL TRANSFER INSTRUCTIONS

Conditional jumps:

- Conditional byte is a two byte instruction: one byte is the opcode of the 'J' condition and the second byte is the value between 00 and FF.
- All conditional jumps are short jumps.
- The range of short jumps is 1 byte = 256 locations (-128 to +127)
- Jumps can be categorized as:
  - Backward jumps (-128 to -1)
  - Forward jumps (0 to +127)
- In backward jump, the second byte is the 2's complement of the displacement value.

CONTROL TRANSFER INSTRUCTIONS

Table 1: Conditional Jump instructions

Mnemonic	Condition Tested	"Jump IF ..."
JA/JNBE	(CF = 0) and (ZF = 0)	above/not below nor zero
JAE/JNB	CF = 0	above or equal/not below
JB/JNAE	CF = 1	below/not above nor equal
JBE/JNA	(CF or ZF) = 1	below or equal/not above
JC	CF = 1	carry
JE/JZ	ZF = 1	equal/zero
JG/JNLE	((SF xor OF) or ZF) = 0	greater/not less nor equal
JGE/JNL	(SF xor OF) = 0	greater or equal/not less
JL/JNGE	(SF xor OR) = 1	less/not greater nor equal
JLE/JNG	((SF xor OF) or ZF) = 1	less or equal/not greater
JNC	CF = 0	not carry
JNE/JNZ	ZF = 0	not equal/not zero
JNO	OF = 0	not overflow
JNP/JPO	PF = 0	not parity/parity odd
JNS	SF = 0	not sign
JO	OF = 1	overflow
JP/JPE	PF = 1	parity/parity equal
JS	SF = 1	sign

CONTROL TRANSFER INSTRUCTIONS

- To calculate the target address , the second byte is added to the IP of the instruction after the jump.

```
1067:0000 B86610          MOV   AX,1066
1067:0003 8ED8     MOV    DS,AX
1067:0005 B90500          MOV   CX,0005
1067:0008 BB0000          MOV   BX,0000
1067:000D 0207     ADD    AL,[BX]
1067:000F 43              INC   BX
1067:0010 49              DEC   CX
1067:0011 75FA     JNZ    000D
1067:0013 A20500          MOV   [0005],AL
1067:0016 B44C     MOV    AH,4C
1067:0018 CD21     INT    21
```

- "JNZ AGAIN" was assembled as "JNZ 000D", and 000D is the address of the instruction with the label AGAIN.
- "JNZ 000D" has the opcode 75 and the target address FA.

CONTROL TRANSFER INSTRUCTIONS

- This is followed by "MOV SUM,AL", which is located beginning at offset address 0013

```
1067:0000 B86610          MOV   AX,1066
1067:0003 8ED8     MOV    DS,AX
1067:0005 B90500          MOV   CX,0005
1067:0008 BB0000          MOV   BX,0000
1067:000D 0207     ADD    AL,[BX]
1067:000F 43              INC   BX
1067:0010 49              DEC   CX
1067:0011 75FA     JNZ    000D
1067:0013 A20500          MOV   [0005],AL
1067:0016 B44C     MOV    AH,4C
1067:0018 CD21     INT    21
```

- The IP value of MOV is 0013, is added to FA to calculate the address of label AGAIN, and the carry is dropped.
- FA is 2's complement of -6, meaning that the address of the target is -6 bytes from the IP of the next instruction.

CONTROL TRANSFER INSTRUCTIONS

- Calculate a forward jump target address by adding the IP of the following instruction to the operand.
- The displacement value is positive in the following example:

```
0005        8A 47 02 AGAIN:   MOV   AL,[BX]+2
0008        3C 61             CMP   AL,61H
000A        72 06             JB    NEXT
000C        3C 7A             CMP   AL,7AH
000E        77 02             JA    NEXT
0010        24 DF             AND   AL,0DFH
0012        88 04   NEXT:     MOV   [SI],AL
```

- "JB NEXT" has the opcode 72, the target address 06 and is located at IP = 000A and 000B.
- The jump is 6 bytes from the next instruction, has IP = 000C.
- Adding gives us 000C + 0006 = 0012, which is the exact address of the NEXT label.

CONTROL TRANSFER INSTRUCTIONS

Unconditional jumps:

- "JMP label" is an unconditional jump in which control is transferred unconditionally to the target location label.
- The unconditional jump can take the following forms:
  - SHORT JUMP
  - NEAR JUMP
    - Direct jump
    - Register indirect jump
    - Memory indirect jump
  - FAR JUMP.

CONTROL TRANSFER INSTRUCTIONS

1) SHORT JUMP - in the format "JMP SHORT label".

- A jump within -128 to +127 bytes of memory relative to the address of the current IP, opcode EB.

2) NEAR JUMP - the default, has the format "JMP label".

- A jump within the current code segment, opcode E9.
- The target address can be any of the addressing modes of direct, register indirect, or memory indirect:

- ***Direct JUMP*** - exactly like the short jump.
- Except that the target address can be anywhere in the segment in the range +32767 to -32768 of the current IP.
- ***Register indirect JUMP*** - target address is in a register.
- In "JMP BX", IP takes the value BX.
- **Memory indirect JUMP** - target address is the contents of two memory locations, pointed at by the register.
- "JMP [DI]" will replace the IP with the contents of memory locations pointed at by DI and DI+1.

3) FAR JUMP - in the format "JMP FAR PTR label". register.

- A jump out of the current code segment.
- IP and CS are both replaced with new values.

CONTROL TRANSFER INSTRUCTIONS

CALL Statements

- Another control transfer instruction is the CALL instruction, which is used to call a procedure.
- CALLs to procedures are used to perform tasks that need to be performed frequently.
- This makes a program more structured.
- The target address could be in the current segment, in which case it will be a NEAR call or outside the current CS , which is a FAR call.

CONTROL TRANSFER INSTRUCTIONS

- When a subroutine is called, control is transferred to that subroutine and the processor saves the IP (and CS in the case of FAR call)and begins to fetch instructions from the new location.
- After finishing the execution of the subroutine, for control to be transferred back to the caller, the last instruction in the called subroutine must be RET (return).

CONTROL TRANSFER INSTRUCTIONS

Sample of Assembly language subroutines is given below:

```
                .CODE
MAIN            PROC FAR          ; This is program entry point
                MOV AX, @DATA
                MOV DS, AX
                CALL SUBR1
                CALL SUBR2
                CALL SUBR3
                MOV AH, 4CH
                INT 21H
MAIN            ENDP
;_____________________________
SUBR1           PROC
                . . . .
                . . . .
                RET
SUBR1           ENDP
;_____________________________
SUBR2           PROC
                . . . .
                . . . .
                RET
SUBR2           ENDP
;_____________________________
SUBR3           PROC
                . . . .
                . . . .
                RET
SUBR3           ENDP
;_____________________________
                END MAIN          ; This is program exit point
```