Program Control Instructions

Hardware Lab

Introduction

- Program Control Instructions direct the flow of a program and allow the flow to change

- Change in flow occurs after a decision, made with CMP or TEST instruction, followed by a conditional jump instruction

THE JUMP GROUP

- Allows programmer to skip program sections and branch to any part of memory for next instruction.

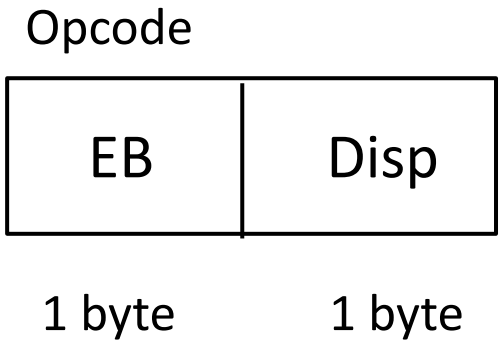
- A conditional jump instruction allows decisions based upon numerical tests.
  - results are held in the flag bits, then tested by conditional jump instructions
- An unconditional Jump instruction does not depend on any condition or numerical tests.

Unconditional Jump (JMP)

- Three types: short jump, near jump, far jump.
- The short and near jumps are often called **intrasegment jumps.**
- Far jumps are called **intersegment jumps.**

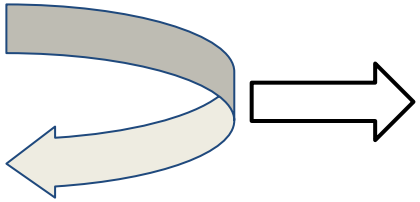
Short Jump

- Called **relative jumps**
- Jump address is not stored with the opcode
- A distance, or displacement, follows the opcode
- The short jump displacement is a distance represented by a 1-byte signed number
- It allows jumps or branches to memory location within +127 and -128 bytes from the address following the jump.




```
0000      XOR BX,BX
0002      START:   MOV AX,1
0005      ADD AX,BX
0007      JMP SHORT NEXT

0020      NEXT:    MOV BX,AX
0022      JMP START
```



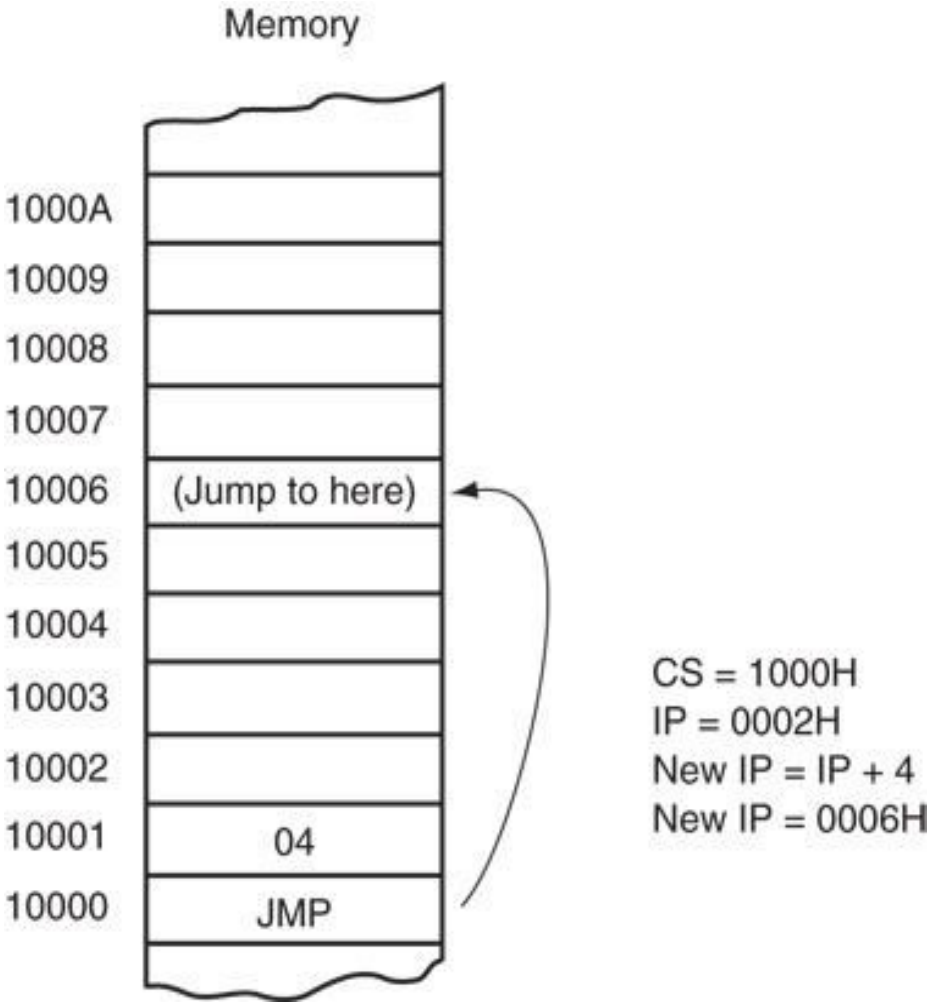
Disp= 0020H – 0009H

= 0017 (17H)

A short jump to four memory locations beyond the address of the next instruction.

Example:

JMP 04H



Near Jump

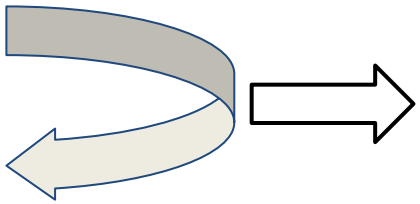
- A near jump passes control to an instruction in the current code segment located within ±32K bytes from the near jump instruction.
- 3-byte instruction with opcode followed by a signed 16-bit displacement.
- Signed displacement is added to the instruction pointer (IP) to generate the jump address.
- can jump to any memory location within the current real mode code segment

Opcode

E9	Disp low	Disp high
1 byte	1 byte	1 byte

```
0000      XOR BX,BX
0002      START:    MOV AX,1
0005      ADD AX,BX
0007      JMP SHORT NEXT

0200      NEXT:     MOV BX,AX
0202      JMP START
```

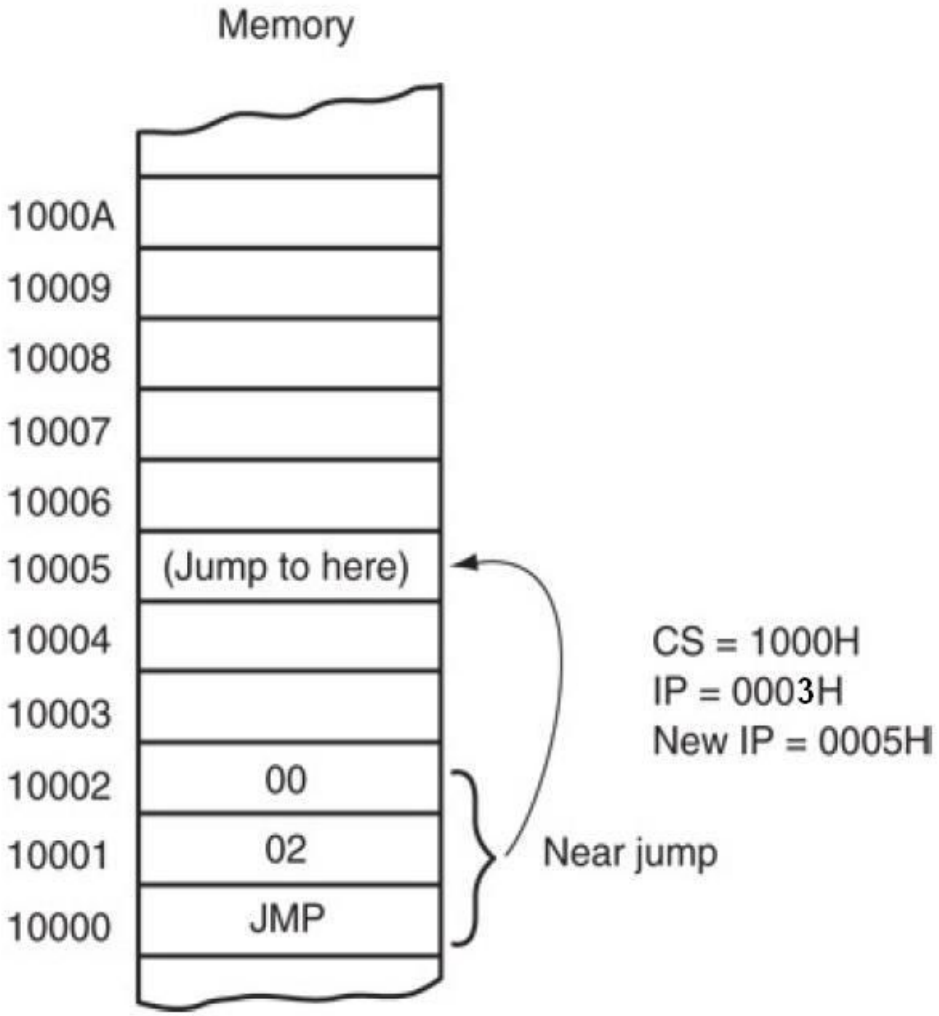


Disp= 0200H – 000AH

= 01F6H

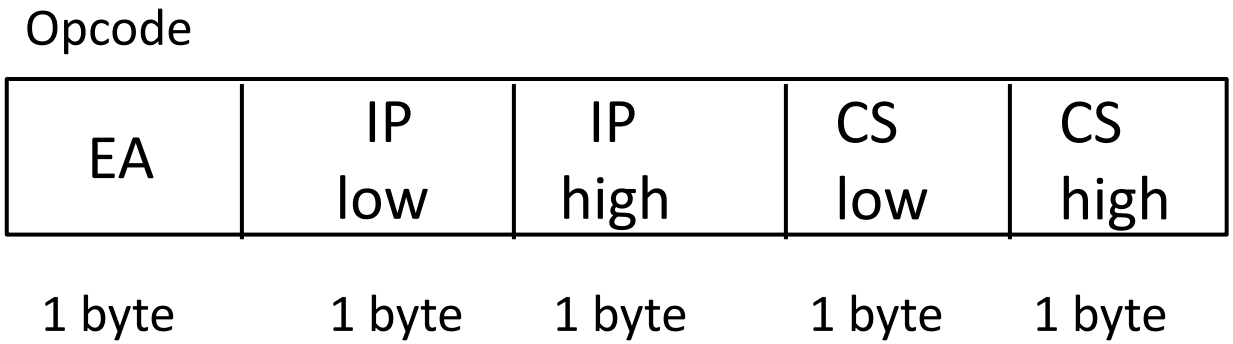
A near jump that adds the displacement (0002H) to the contents of IP.

Example:
JMP 0002H



Far Jump

- 5-byte instruction
- Bytes 2 and 3 of this instruction contain the new offset address
- Bytes 4 and 5 contain the new segment address
- offset address (16 or 32 bits) - contains the offset address within the new code segment



Far Jump

- Far jump instruction appears with the FAR PTR directive
- OR define a label as a *far label.*
- EXTRN UP:FAR -> defines the label UP as a far label
- A label is far only if it is external to the current code segment or procedure.
- The JMP UP instruction in the example references a far label.

Example:

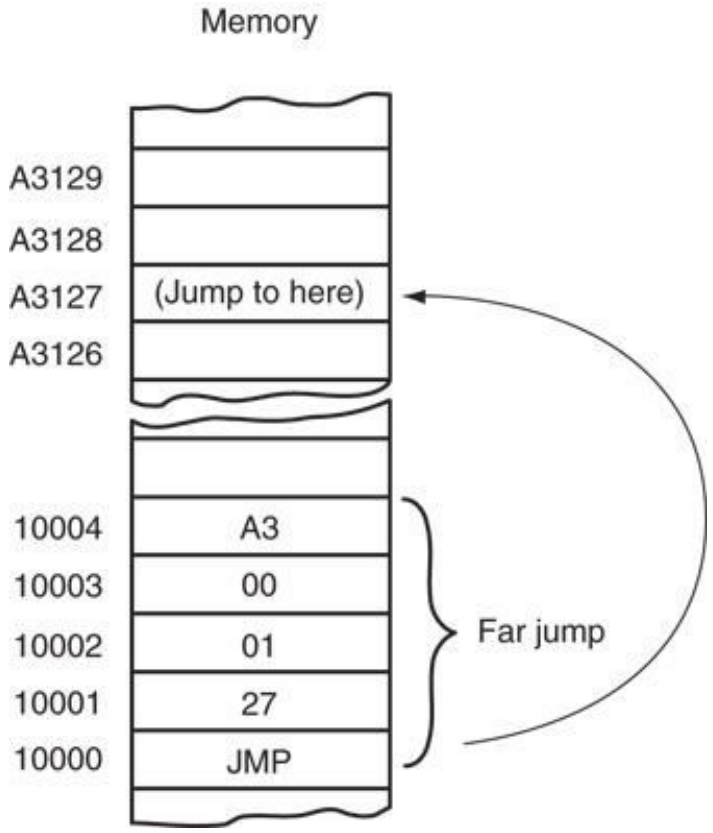
```
EXTRN UP:FAR
      XOR BX,BX
START: ADD AX,1
 JMP NEXT

NEXT:  MOV BX,AX
 JMP FAR PTR START
JMP UP
```

A far jump instruction replaces the contents of both CS and IP with 4 bytes following the opcode.

IP= 10005
New offset= 0127
New CS=A300
New IP= A3000+0127=A3127

Example:
JMP 0127: A300



Jumps with Register Operands

- Jump can use a 16- or 32-bit register as an operand.
- automatically sets up as an **indirect jump**
- allows a jump to any location within the current code segment
- Register content (address of the jump) is transferred directly into IP
- JMP AX - copies the contents of AX register into the IP

Conditional Jumps

- Always short jumps in 8086 - 80286.
- In 80386 and above, conditional jumps are short or near jumps
- Jump to any location within the current code segment.
- Conditional jump instructions test flag bits:
  - sign (S), zero (Z), carry (C), parity (P), overflow (O)

- Condition = true branch to label
- Condition = false execute the next sequential step in program

TABLE 6–1 Conditional jump instructions.

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)

Conditional Jumps

- To compare signed numbers, use
  - JG, JL, JGE, JLE, JE, and JNE instructions.
- To compare unsigned numbers, use
  - JA, JB, JAB, JBE, JE, and JNE instructions.
- Remaining conditional jumps test individual flag bits, such as overflow and parity.

Conditional Set

- 80386 onwards
- set a byte to either 01H or clear a byte to 00H, depending on the outcome of the condition under test
- useful where a condition must be tested at a later point in the program.
- SETNC MEM instruction - places 01H into memory location MEM if carry is cleared, and 00H into MEM if carry is set.

TABLE 6–2 Conditional set instructions.

Assembly Language	*Tested Condition*	*Operation*
SETA	Z = 0 and C = 0	Set if above
SETAE	C = 0	Set if above or equal
SETB	C = 1	Set if below
SETBE	Z = 1 or C = 1	Set if below or equal
SETC	C = 1	Set if carry
SETE or SETZ	Z = 1	Set if equal or set if zero
SETG	Z = 0 and S = 0	Set if greater than
SETGE	S = 0	Set if greater than or equal
SETL	S != O	Set if less than
SETLE	Z = 1 or S != O	Set if less than or equal
SETNC	C = 0	Set if no carry
SETNE or SETNZ	Z = 0	Set if not equal or set if not zero
SETNO	O = 0	Set if no overflow
SETNS	S = 0	Set if no sign (positive)
SETNP or SETPO	P = 0	Set if no parity or set if parity odd
SETO	O = 1	Set if overflow
SETP or SETPE	P = 1	Set if parity or set if parity even
SETS	S = 1	Set if sign (negative)

LOOP

- A combination of a decrement CX and the JNZ conditional jump.
- In 8086 - 80286 LOOP decrements CX.
  - if CX != 0, jumps to the address indicated by the label
  - If CX = 0, the next sequential instruction executes

LOOP

Example:

```
MOV CX, 25H

MOV AX, 1H

MOV BX, 4H

XXX : ADC AX, BX

      ADD BX, 3H

      DEC CX

      JNZ XXX // jump if result (value of CX) not zero
```

Conditional LOOPs

- LOOPE (loop while equal) instruction
  - jumps if CX != 0 while an equal condition exists.
  - will exit loop if the condition is not equal or CX register decrements to 0

- LOOPNE (loop while not equal)
  - jumps if CX != 0 while a not-equal condition exists.
  - will exit loop if the condition is equal or the CX register decrements to 0

Thank You