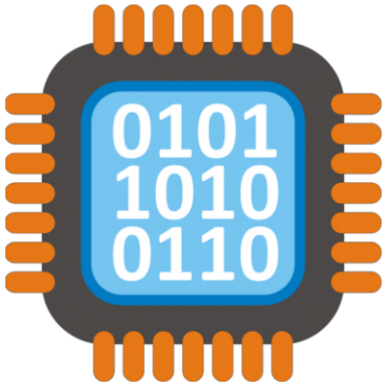
Secure Assembly Coding

Week # 10 Lectures



Dr. Qasem Abu Al-Haija
Department of Cybersecurity,

Branch Instructions

8086 Instruction Set

Group 4 : Unconditional Transfer Instructions

- **Used to transfer control to: Intra-segment or Inter-segment.**

CALL reg/mem/disp 16	Call subroutine
RET or RET disp 16	Return from subroutine
JMP disp8/disp 16 /reg16/mem16	Unconditional jump

- **CALL Instruction.**

—Intra-segment CALL: IP changes, CS is fixed, EX: **CALL NEAR PROC**.
—Inter-segment CALL: Both IP & CS are changed, **EX: CALL FAR PROC**.

- **RET instruction.**

—Placed at the end of the subroutine to transfer control back to the main program.

- **JMP Instruction.**

—Intra-segment JMP: IP changes, CS is fixed, EX: **JMP START.**
—Inter-segment JMP: Both IP & CS are changed, **EX: JMP FAR BEGIN.**

8086 Instruction Set

Group 4 : Unconditional Transfer Instructions

```
CODE        SEGMENT
            ASSUME      CS:CODE, DS:DATA, SS:STACK
            ------------
            ------------
            CALL   MULTI
            ------------
            ------------
            HLT
MULTI       PROC         NEAR
            ------
            ------
            RET
MULTI       ENDP
CODE        ENDS
```

Example of: Intra-segment CALL

```
ORG 100h
.CODE
MOV AX, 2
MOV BX, 2
JMP LABEL_SUB
ADD AX, BX   ;this instruction will never execute
LABEL_SUB:
            SUB AX, BX
RET
```

Example of: JMP destination_label

8086 Instruction Set

Group 4 : Unconditional Transfer Instructions

Example of: Inter-segment CALL

```
CODE        SEGMENT
            ASSUME      CS:CODE, DS:DATA, SS:STACK
            ------------
            ------------
            CALL  MULTI
            ------------
            ------------
            HLT
CODE        ENDS
SUBR        SEGMENT
MULTI       PROC        FAR

            ASSUME      CS:SUBR
            ------
            ------
            RET
 MULTI      ENDP
 SUBR       ENDS
```

8086 Instruction Set

Group 5: Conditional Branch Instructions

- All 8086 conditional branch instructions use 8 bit signed disp.
- Range of -128 to +127, with 0 being positive.

If condition is true,
then IP ← IP + disp8,
otherwise IP ← IP + 2 and execute next instruction.

- Two types of conditional branch instructions.
  - **—Using COMPARE instruction.**
    - Equal, above, below, less than, or greater than.
    - Can be used for both signed and unsigned.
  - **—Setting of flags (to check the result).**
    - Zero or nonzero, Positive or negative, Did or did not produce a carry, Did or did not produce parity, Did or did not cause overflow.
    - Instructions for this: JZ, JNZ, JS, JNS, JC, JNC, JP, JNP, JO, JNO.

8086 Instruction Set

Group 5: Conditional Branch Instructions

Jumps for Unsigned Numbers

Instructions	Meaning	Condition
JA JNA	Jump if above >	CF = 0 and ZF = 0
JAE JNAE	Jump if above or equal ≥	CF = 0
JB JNB	Jump if below <	CF = 1
JBE JNBE	Jump if below or equal ≤	CF = 1 or ZF = 1
JC JNC	Jump in carry	CF = 1

8086 Instruction Set

Group 5: Conditional Branch Instructions

Example:

Instructions	UNSIGNED FLAGS	SATISFIED JUMP
MOV AL, 10	-	
MOV BL, 5	-	
CMP AL, BL	ZF=0. CF=0	JA, JAE, JNB, JNC, JNZ
CMP BL, AL	ZF = 0. CF = 1	JB, JC, JNA

8086 Instruction Set

Group 5: Conditional Branch Instructions

Jumps for Signed Numbers

Instructions	Meaning	Condition
JG	Jump if greater	ZF = 0 and SF = OF
JGE	Jump if greater or equal	SF=OF
JL	Jump if less	SF<>OF
JLE	Jump if less or equal	ZF=1 OR SF<>OF
JS	Jump on signe flag	SF=1
JNS	Jump on not signe flag	SF=0
JO	Jump on over flow	OF=1
JNO	Jump on not over flow	OF=0

8086 Instruction Set

Group 5: Conditional Branch Instructions

8086 Conditional Branch Instructions Affecting Individual Flags	
JC disp8	JUMP if carry, i.e., CF = 1
JNC disp8	JUMP if no carry, i.e., CF = 0
JP disp8	JUMP if parity, i.e., PF = 1
JNP disp8	JUMP if no parity. i.e., PF = 0
JO disp8	JUMP if overflow, i.e., OF = 1
JNO disp8	JUMP if no overflow, i.e., OF = 0
JS disp8	JUMP if sign, i.e., SF = 1
JNS disp8	JUMP if no sign, i.e.. SF = 0
JZ disp8	JUMP if result zero, i.e.. ZF = 1
JNZ disp8	JUMP if result not zero, i.e., ZF = 0

8086 Instructions To Be Used after CMP A, B;			
Signed "a" and "b"		*Unsigned "a" and "b"*	
JGE disp8	if a ≥ b	JAE disp8	if a ≥ b
JL disp8	if a < b	JB disp8	if a < b
JG disp8	if a > b	JA disp8	if a > b
JLE disp8	if a ≤ b	JBE disp8	if a ≤ b

8086 Instruction Set

Group 5: Conditional Branch Instructions

Example:

```
        MOV   AX, 1000H
        MOV   DS, AX              ;Initialize DS
        MOV   BX, 2000H
        MOV   CX, 3000H
AGAIN:  MOV   WORD PTR[BX], 0000H
        INC   BX
        INC   BX
        CMP   CX, BX
        JGE   AGAIN
```

JGE treats CMP operands as twos complement numbers.

Example:

```
ax = 2;
if ( ax != bx )
{
ax = ax + 1 ;
}
bx = bx + 1 ;
```

```
mov ax, 2 ; ax = 2
sub ax, bx ; ax = 2 - bx
jz nextl ; jump if (ax-bx) == 0
      inc ax ; ax = ax + 1
nextl:
      inc bx
```

Iteration Instructions

8086 Instruction Set

Group 8 : Iteration Control Instructions

All these instructions have relative addressing modes.

8086 Iteration Control Instructions	
LOOP disp8	Decrement CX by 1 without affecting flags and branch to label if CX ≠ 0; otherwise, go to the next instruction.
LOOPE/LOOPZ disp8	Decrement CX by 1 without affecting flags and branch to label if CX ≠ 0 and ZF = 1; otherwise (CX=0 or ZF=0), go to the next instruction.
LOOPNE/LOOPNZ disp8	Decrement CX by 1 without affecting flags and branch to label if CX ≠ 0 and ZF = 0; otherwise (CX=0 or ZF=1), go to the next instruction.
JCXZ disp8	JMP if register CX =0.

Example:

```
        DEC   SI
        MOV   CX,50                ; Initialize CX with array count
BACK:   INC   SI                   ; Update pointer
        CMP   BYTE PTR[SI],00H     ; Compare array element with 00H
        LOOPE BACK
```

See other examples in the separate ppt file uploaded into Moodle

More Example_3

Main Sources for these slides

- *K. R. Irvine. Assembly Language for x86 Processors, 8th edition, Prentice-Hall (Pearson Education), June 2019. ISBN: 978-0135381656.*

- *B. Dang, A. Gazet, E. Bachaalany. Practical Reverse Engineering: x86, x64, ARM, Windows® Kernel, Reversing Tools, and Obfuscation. John Wiley & Sons, June 2014. ISBN: 978-1-118-78731-1*

- *Qasem Abu Al-Haija, "Microprocessor Systems", King Faisal University, Saudi Arabia*

- *Ghassan Issa, "Computer Organization", Petra University, Jordan.*

Thank you