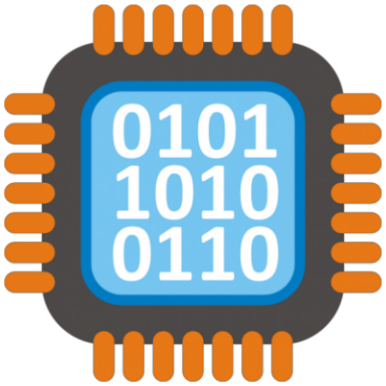
Secure Assembly Coding

Week # 6 Lectures



Dr. Qasem Abu Al-Haija,
Department of Cybersecurity

Introduction to assembly Instructions

Simple Instructions

- MOV (Assignment)
- INC (Add 1)
- DEC (Subtract 1)
- ADD (Add two numbers)
- SUB (Subtract two numbers)
- CMP (Compare two numbers)
- JMP (Go to)
- JNZ (Go to if results is not zero)
- JZ (Go to if results is zero)

Instructions Format

- Two Operand Instructions

  General Form:

  <Instruction> <Target Operand>, <Source Operand>

  Examples

```
MOV AX, 5          ; ASSIGN AX THE VALUE 5
MOV DX, AX         ; ASSIGN DX WHATEVER VALUE IN AX
MOV NUMS, 2        ; STORE 2 IN VARIABLE NUMS
ADD CX, 2          ; ADD 2 TO THE VALUE OF CX
CMP AX, 5          ; COMPARE THE VALUE OF AX WITH 5
SUB AX, BX         ; AX = AX - BX
```

One operand instructions

- **General form:**
  <Instruction> <Destination>
  Destination: reg., variable

Examples

```
INC AX              ; AX = AX +1
INC NUMS            ; NUMS = NUMS + 1
DEC BX              ; BX = BX –1
JMP   LABEL1        ; GO TO LABEL1
JNZ   LABEL1        ; DON'T JUMP IF RESULTS IS ZERO
JZ    LABEL1        ; JUMP IF REULTS IS ZERO
```

General Assembly Language Rules

1. **Operands must be equal size at all times**
   - **Mov Op1 (8-bit), Op2 (8-bit) …… Ok**
   - **Mov Op1 (16-bit), Op2 (16-bit) …… OK**
   - **Mov Op1 (16-bit), Op2 (8-bit) Wrong…. Wrong**

General Assembly Language Rules

- **An instruction can not refer or use two memory locations. The two operands can not be memory locations. Its ok to use a memory location with a register or a constant**

- Mov num1, num2 .... Wrong
- **MOV AX, NUM1 ..... OK**
- **MOV NUM2, AX ..... OK**
- Mov num1, [BX] ..... Wrong
- Mov num1, 10 ..... OK

General Assembly Language Rules

- **The destination of any instruction should not be a constant**
  - **Mov 10, num …… Wrong**
  - **Inc 10 …… Wrong**

General Assembly Language Rules

- **One of the operands of any instruction should specify the size (8 or 16 bit) of the instruction**
  - **Mov [BX], 10 ..... Wrong**
  - **Inc [BX]..... Wrong**
  - **Inc Byte PTR [BX].....OK**
  - **Add Word PTR [BX], 10 .....OK**
  - **Mov [BX], AL..... OK**
- Note: [BX] may refer to an 8-bit or 16-bit location. It does not really specify the size

8086 Instruction Set

8086 Instruction Set

- **Approximately: 117different instructions with 300 op-codes, divided in to 10-groups:**

❖ **Group 1: Data Transfer Instructions.**
— MOV, PUSH, POP, XCHG, XLAT, IN, OUT, LEA, LDS, LES, LSS.

❖ **Group 2: Arithmetic Instructions.**
— ADD, ADC, SUB, SBB, INC, DEC, NEG, CMP, MUL, IMUL, DIV, IDIV, DAA, DAS, AAA, AAS, AAD, AAM.

❖ **Group 3: Bit Manipulation Instructions (logical).**
— AND, OR, XOR, NOT, TEST, SHL, SHR, ROL, ROR.

❖ **Group 4: Unconditional Transfer Instructions.**
— JMP, CALL, RET

❖ **Group 5: Conditional Branch Instructions.**
— JE, JZ, JNE,JNZ, JL, JNGE, JNL, JGE, JG, JNLE, JNG, JLE, JB, JNAE, JNB, JAE, JA, JNBE, JNA, JBE, JP, JPE, JNP, JPO

❖ **Group 6: Iteration Control Instructions.**
— LOOP, LOOPE, LOOPZ, LOOPNE, LOOPNZ, JCXZ

❖ **Group 7: String Instructions.**
— MOVSB, MOVSW, CMPSB, CMPSW, LODSB, LODSW, STOSB, STOSW, SCASB, SCASW, REP, REPE, REPNE.

❖ **Group 8: Interrupt Instructions.**
— INT N, INTO, IRET

❖ **Group 9: Flags manipulation Instructions.**
— LAHF, SAHF, PUSHF, POPF, CMC, CLC, STC, CLD, STD, CLI

❖ **Group 10: Processor control Instructions.**
— NOP, WAIT, ESC, LOCK, and HLT.

STOP and Go Over 8086 Instruction set Appendix.

Group 1:

Data Transfer Instructions.

MOV, PUSH, POP, XCHG,

XLAT, IN, OUT, LEA,

LDS, LES, LSS.

8086 Instruction Set

Group 1 : Data Transfer Instructions

- **MOV instruction.**

— Copies a word or byte of data from a specified source to a specified destination.

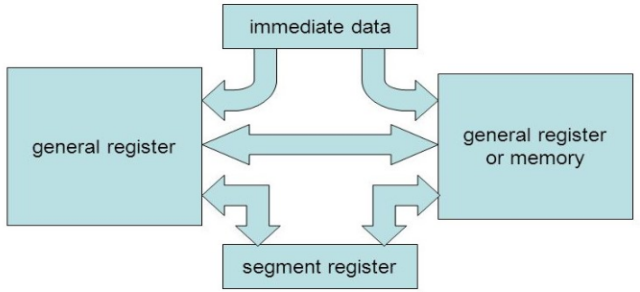
— The destination can be a register or a memory location.

— The source can be a register or a memory location or an immediate number.

— General format is: MOV Destination, Source

'mov'

- It's the most frequently-used instruction



- **Examples of MOV instruction.**
  - **MOV CX, DX** ; Copies 16-bit contents of DX into CX
  - **MOV AX, 2025H** ; Moves immediate data 2025 to AX register
  - **MOV CH, [BX]** ; BX = 0050H, DS = 2000H, Mem Loc (20050) = 08
    ; 8-bit contents of memory location DS+BX will be transferred to CH register, memory location is 20000 + 00050 = (20050)H ➔ CH will contain 08H
  - **MOV START [BP], CX** ; CX = 5009H, BP = 0030H, SS = 3000H, START = 06H
    ;16-bit contents of register CX will be stored in memory location SS+START+BP = 30000 + 00030 + 06 =(30036)H = 09H(CL) and memory location (30037) = 50H (CH).

8086 Instruction Set

Group 1 : Data Transfer Instructions

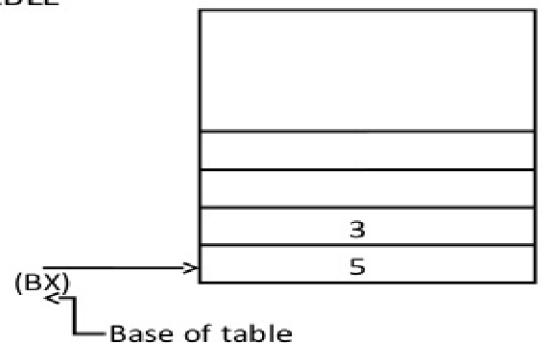
- **XCHG instruction.**
  - Exchanges the register contents with the contents of memory location.
  - It cannot exchange directly the contents of two memory locations.
  - The source and destination must both be words or must both be bytes.
  - The segment registers cannot be used in this instruction.
- **Examples : XCHG AL, BL / XCHG CX,BX / XCHG AL, [BX].**

- **XLAT instruction.**
  - Used to replace the byte in AL with a byte from user's table, addressed by BX.
  - Original value of AL is index into translate table.

Mnemonic	Meaning	Format	Operation	Flags
XLAT	Translate	XLAT	((AL)+(BX)+(DS)0) → (AL)	None

```
MOV  BX, OFFSET TABLE
MOV  AL, 00H
XLAT
(AL) ← 5

AL← [BX+AL]
```



Another Example:

(assume any value)

MA =(DS)0 + (BX) + (AL)

$= 0300O_{16} + 0100_{16} + 0D_{16}$

$= 030OD_{16}$ ($030OD_{16}$) (AL)

8086 Instruction Set

Group 1 : Data Transfer Instructions

- **IN instruction**
  - Copies data from a port to AL or AX register (Direct or Indirect/variable port).
  - If 8-bit port is read, data is stored in AL, if 16 bit port is read, data is stored in AX.

  - **Examples :**
  - **IN AL, 38H** ; Input data from port 38H to AL register
  - **IN AL, DX** ; Input 8-bit data from 8-bit port specified by DX

IN AL, imm byte
IN AL, DX
IN AX, imm byte
IN AX, DX

- **OUT instruction**
  - Transfer data from AL or AX register to a port (Direct or Indirect/variable port).
  - If 8-bit is transferred, data is taken from AL, if 16 bit, data is taken from AX.

  - **Examples :**
  - **OUT 38H, AL** ; Output data from AL register to 38H
  - **OUT DX, AX** ; Output AX data in to port specified by DX register

OUT imm byte, AL
OUT imm byte, AX
OUT DX, AL
OUT DX, AX

8086 Instruction Set

Group 1 : Data Transfer Instructions

- **LEA instruction (**Load Effective Address**)**
  - Determines the offset address of a variable or memory location named as the source and puts this offset address in the indicated 16-bit register.
  - The general format of LEA instruction is: LEA register, source.
  - **Examples :**
  - **LEA BX, COST** ; BX= Offset address of COST in data segment where COST is ; the name assigned to a memory location in data segment.
  - **LEA CX, [BX][SI]** ; CX= (BX)+(SI) (content of BX and SI respectively).

- **LDS instruction (**Load register and DS with words from memory**)**
  - Copies a word from memory location specified in the instruction into register and then copies a word from the next memory location into the DS register.
  - LDS is useful for initializing SI and DS registers at the start of a string before using one of the String instructions.
  - **Examples :**
  - **LDS SI,[2000H]** ; Copy the contents of memory word at offset address 2000H in ; data segment to SI register and the contents of memory word ; at offset address 2002H in data segment to DS register.

- **LES, LSS instructions**
  - Similar to LDS instruction except that instead of DS register, ES and SS registers are loaded respectively along with the register specified in the instruction.

8086 Instruction Set

Group 1 : Data Transfer Instructions

- **PUSH instruction.**
  - Used to store a word from a register or a memory location into stack.
  - SP is decremented by 2 after execution of PUSH.
  - **Example: PUSH CX, PUSH DS**

- **POP instruction.**
  - Copies the top word from stack into a destination specified in the instruction.
  - The destination can be a GPR, a segment register or a memory location.
  - SP is incremented by 2 after execution of POP to point to the next word in stack.
- **Examples : POP CX / POP DS / POP [SI].**

Example: if BX, DX, and SI are PUSHed:

```
PUSH    BX
PUSH    DX
PUSH    SI
```

Then: they must be POPped using:

```
POP     SI
POP     DX
POP     BX
```

8086 Instruction Set-
Group 1: Data Transfer Instructions

Example of PUSH instruction: PUSH [BX] , Assume that:

DS = 2000H, BX = 0200H, SP = 3000H, SS = 4000H, (20200) = 0120H

BEFORE					AFTER				
SP	3000	Memory locations	20200	20	SP	2FFE	Memory locations	20200	20
DS	2000		20201	01	DS	2000		20201	01
SS	4000				SS	4000			
BX	0200	Memory locations	42FFE	xx	BX	0200	Memory locations	42FFE	20
			42FFF	xx				42FFF	01

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