

Introduction to Assembly Language & Programming

EN202 - Computer Organization

By-
Dilum Bandara
dilumb@cse.mrt.ac.lk

Outline

- ▣ Levels of Programming
- ▣ Assembly as a programming language
- ▣ CPU Registers
- ▣ Addressing
- ▣ Programming in Assembly
- ▣ Instructions
- ▣ Examples

EN202

2

Bit about History...

- ▣ 1st generation of microprocessors
 - Intel 8008 in 1970s
- ▣ Then 2nd & other generations followed
 - 8080, 8086, 80286, 80386, 80486
 - Pentium, Pentium II, III, IV
- ▣ All of these were general purpose processors
- ▣ Each family contains its unique set of instructions

EN202

3

Levels of Programming

1. Machine Language
2. Assembly Language
3. High-level Languages

EN202

4

Machine Language

- ▣ What the machine can understand
- ▣ Ones & Zeros 100111010101110.....
- ▣ Individual instructions that the processor executes one at a time
- ▣ Example : using machine language in practical 4

EN202

5

Assembly Language

- ▣ Symbolic instruction that we can understand
- ▣ Simple atomic instructions
- ▣ Maps directly into machine language (1-to-1)
- ▣ Is designed for a family of microprocessors

```
MOV AX, BX
ADD AX, 5A
INC BX
JMP 100
```

Compile

```
01010011110
11010110110
10011101010
01010100110
```



EN202

High-level Languages

- Are the modern programming languages
- C/C++, VB, Java, C#.....
- Much more simple to understand & remember
- Keywords in high-level languages corresponds to many Assembly instructions

EN202

7

Assembly vs High-level

- Simpler than the Machine language
- Require less memory & execution time
- Can perform highly technical &/or complex tasks

However:

- Specific to a processor family
- Harder to remember & understand
- Knowledge in Computer Architecture is necessary.

EN202

8

Registers

- Registers in a microprocessor
 - General purpose registers
 - Special purpose registers

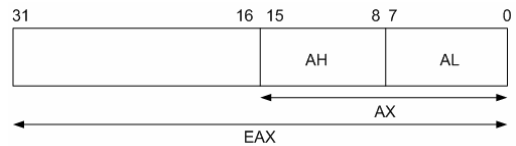
Note : We will only consider processors which support x86 assembly instructions

EN202

9

General Purpose Registers (GPR)

- Are the registers which are available to the programmer
 - AX, BX, CX, DX – 16-bit version
 - EAX, EBX, ECX, EDX – 32-bit version



EN202

10

GPR cont...

AX	Known as the <i>Primary Accumulator</i> is used for operations involving inputs/outputs & arithmetic.
BX	Known as the <i>Base</i> register is used to hold the index in addressing. Can also be used in computation.
CX	Known as the <i>Count</i> register. Used to control loops. Can also be used in computation.
DX	Known as the <i>Data</i> register. Used for input/output operations.

EN202

11

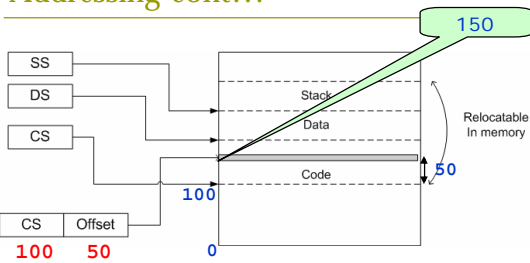
Addressing

- In a program you need to address your data, program code & stack in the memory
- Addressing can be:
 - Absolute address – direct addressing
 - Relative address – indirect addressing
- Segments
 - Special areas in memory that is used to hold code, data & stack

EN202

12

Addressing cont...



EN202

13

Segment + Offset addressing

- We indicate the address using the segment boundary & offset
- Example 1:
 - What is the actual memory location if DS register is $0x03E0$ & offset is $0x32$.

DS	03E0
Offset	<u>32</u> +
Address	412

EN202

14

Pointer Registers

- Are used to hold the offset value
- Example 2:
 - What is the actual memory location if CS register is $0x39B0$ & Instruction Pointer (IP) register is $0x514$.

DS	39B0
Offset	<u>514</u> +
Address	3EC4

EN202

15

Programming in Assembly

- Statement
 - An assembly program consists of set of statements
 - Format:

[Identifier]	Operation	[Operand(s)]	[; Comment]
L30:	MOV	AX, 6	; AX ← 0x06

EN202

16

Assembly instructions

- Assembly instructions can be categorised as:
 - Data transfer instructions
 - Arithmetic instructions
 - Logic operations
 - Bit shifting instructions
 - Transfer instructions (jump instructions)
 - And many more...

EN202

17

Data transfer instructions

- Used to transfer data from one location to another.
 - Register to Register, Register to Memory, Memory to Register, Immediate value to Register
 - MOV – MOV AX, 3 ; AX ← 0x03

EN202

18

Arithmetic instructions

- Used in arithmetic operations
 - ADD – ADD AX, BX ; AX ← AX + BX
 - ADD – ADD AX, 3 ; AX ← AX + 3
 - SUB – SUB BX, AX ; BX ← BX - AX
 - MUL – MUL CX ; AX ← AX * CX
 - INC – INC AX ; AX ← AX + 1
 - DEC – DEC AX ; AX ← AX - 1

EN202

19

Arithmetic instructions cont...

- Example 3:
 - Write an assembly program to add 5 & 10
- Steps:
 - How many registers?
 - What are the registers?
 - What are the instructions?
 - MOV – MOV AX, 5 ; AX ← 0x05
 - ADD – ADD AX, A ; AX ← AX + A

EN202

20

Arithmetic instructions cont...

- Example 2:
 - Write an assembly program to multiply 3 & 4
- Steps:
 - How many registers?
 - What are the registers?
 - What are the instructions?
 - MOV – MOV AX, 3 ; AX ← 0x3
 - MUL – MUL CX ; AX ← AX * CX

EN202

21

Logic operations

- Used in logic operations
 - AND – AND AL, 3 ; AL ← AL & 0011
 - NOT – NOT AH ; AH ← ~AH
 - XOR – XOR CX, 09 ; CX ← CX Å 1001

EN202

22

Logic operations cont..

- Example 7:
 - Write an assembly program to convert a given character from uppercase to lowercase & vice versa.
 - If we consider ASCII this can be achieved by changing the 5th bit.
 - A = 65 = 0x41 = 0 1 0 0 0 0 1
 - a = 97 = 0x61 = 0 1 1 0 0 0 1
 - Get the XOR with 00100000 = 32 = 0x20

EN202

23

Transfer instructions (jump instructions)

- Can be used to jump here & there within the program
- Can be used to control loops
 - JMP – JMP L20 ; jump to line20
 - JZ – JZ L20 ; jump to line20 if zero
 - JNZ – JNZ L20 ; jump to line20 if not zero
 - JE
 - CMP BX, 5 ; check whether BX == 5
 - JE L20 ; if equal go to line 20

EN202

24

Transfer instructions cont...

- Example 8:
 - Write a program to calculate the total of all the integers from 1 to 10
- High-level program:

```
int total = 0;
For (int i=1 ; i<=10; i++)
{
    total += i;
}
```

EN202

25

Transfer instructions cont...

- Steps:
 - Are there any conditions/loops?
 - Use CX register
 - How many registers?
 - What are the registers?
 - What are the instructions?
 - ADD
 - Increment/decrement instruction – INC, DEC
 - Loop controlling instruction – JZ, JNZ, JE

EN202

26

Transfer instructions cont...

```
int total = 0;
For (int i=10 ; i!=0; i--)
{
    total += i;
}
```

```
MOV AX, 0      ; AX ← 0
MOV CX, 0A     ; CX ← 10 (0x0A)
Loop: ADD AX, CX ; AX ← AX + CX
      DEC CX    ; CX ← CX - 1
      JNZ Loop  ; if zero flag is not set go to Loop
```

EN202

27

Beyond the lecture

- Will do more examples in the tutorial
- Exam
 - What is in the note is enough
- For future – Level III/IV
 - You need to learn more on your self

EN202

28