EE 213 Computer Organization and Assembly Language

Week # 1, Lecture # 3

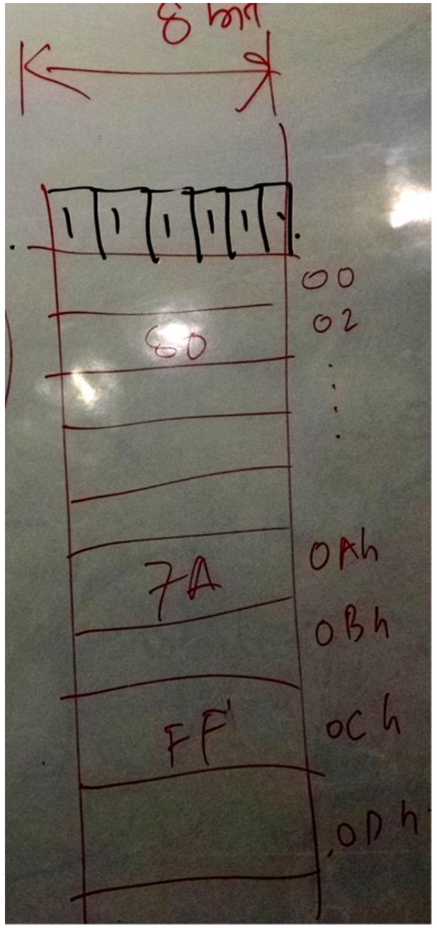
16th Dhu‘l-Hijjah, 1439 A.H

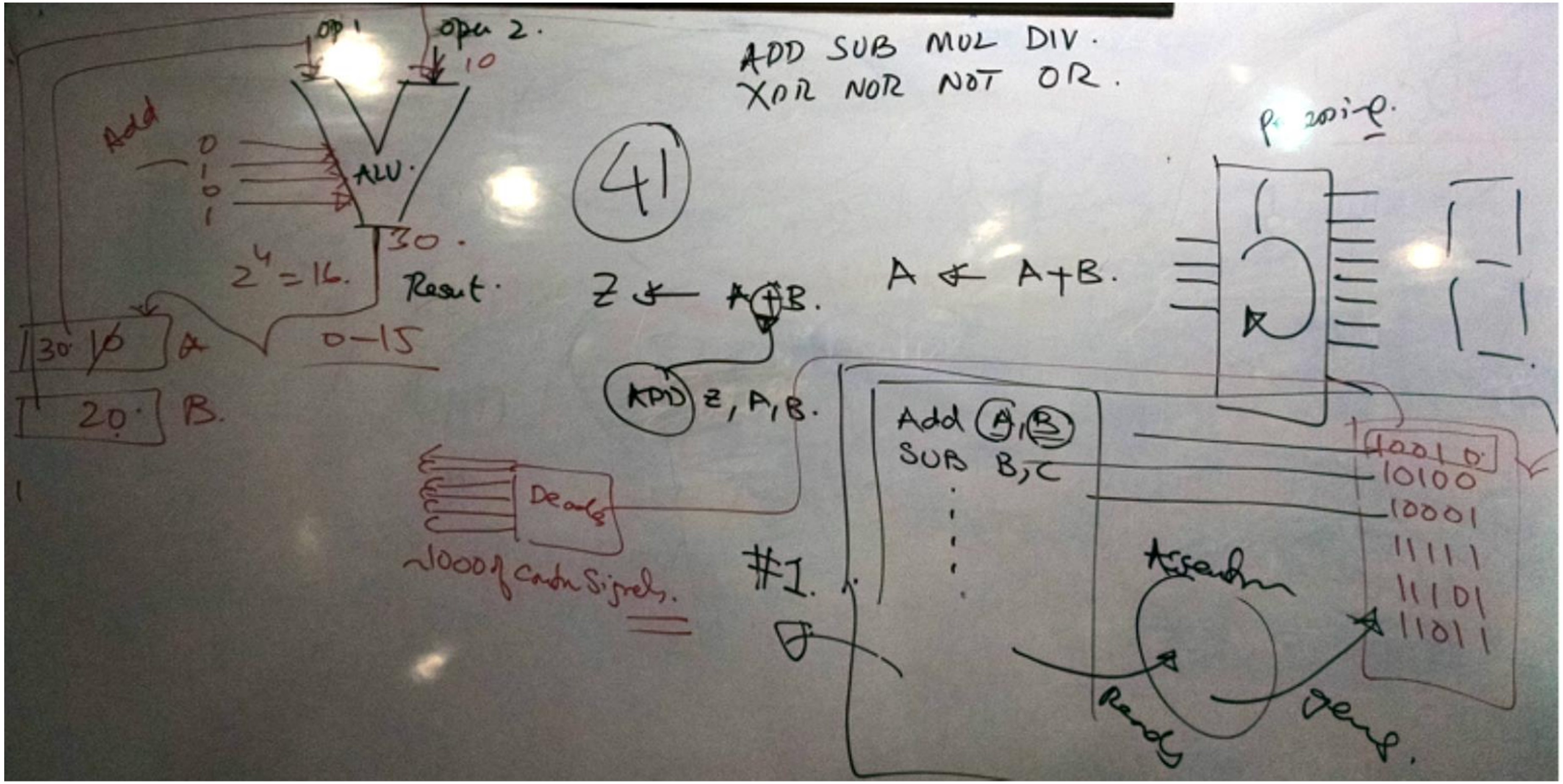
31st August 2018

Revision of Topics from Previous Lecture

- Design of programmable digital circuit
- Von Neumann Architecture
  - Input / Output
  - CPU (Central Processing Unit)
    - Control Unit (CU)
    - Arithmetic Logical Unit (ALU)
  - Memory
- Instructions = Operations + operands + Rules
- Instructions are in binary code also called machine code as it is readable by processor.
- Machine code is cumbersome to read and understand due to long binary digits.
- Assembly code is used instead of machine code. Assembly code -> Assembler (just like compiler but process assembly code) -> linker -> machine code.
- ISA describes the functionality of the micro-architecture to a system programmer. It defines all instructions completely and without ambiguity.
- Processor contains ALU which performs all arithmetic and logic operations.
- Control Unit decodes machine code and generate control signals for ALU and other elements of microarchitecture.
- Memory
  - Processor registers
  - Caches
  - External memory (RAM)
  - Usually shown as an array of numbers with addresses.
- OS load machine code (and constant data) from the .exe file into memory.
- Processor reads instruction one-at-a-time, decode instructions, reading memory operands, execute and save results in memory.
- Data can be read from input devices and written to output devices (keyboards, touch pads and screens, network cards, USB devices, HDMI ports, etc.)

Lecture # 2 Whiteboard Snaps

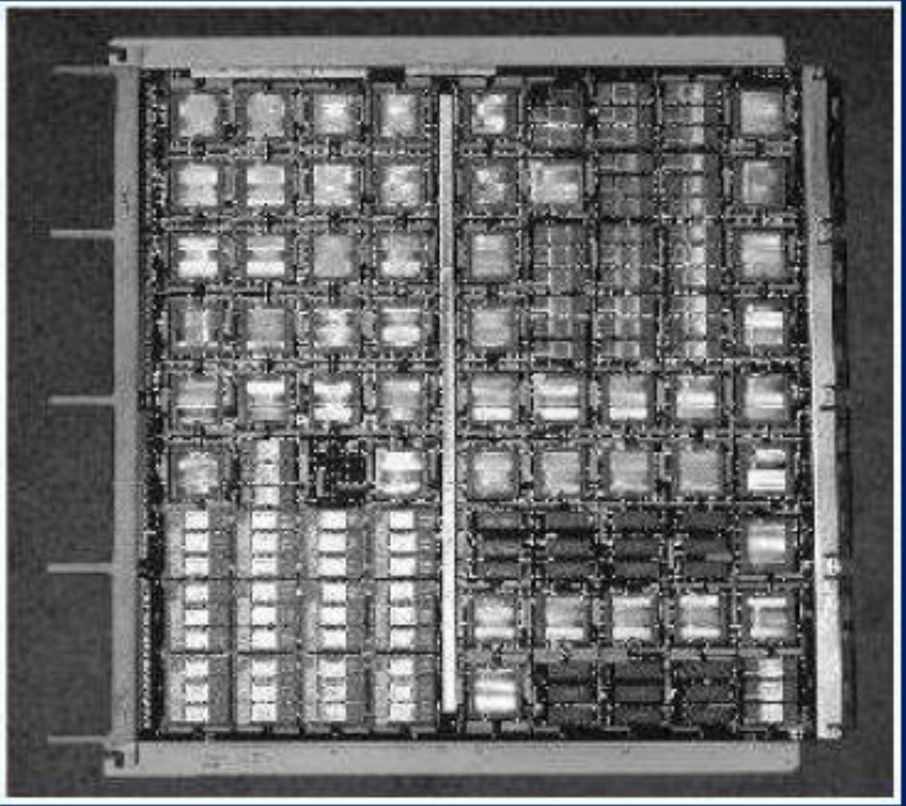




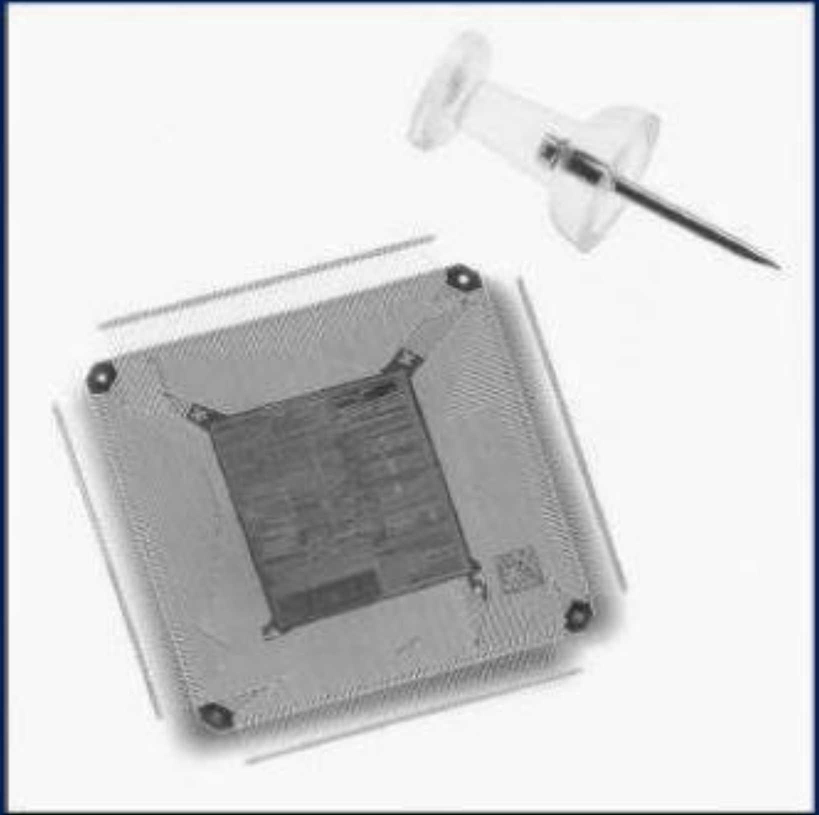
Today's Topics

- Input, Storage and Output devices
- Von Neumann architecture execution
  - Control Unit
  - ALU
  - Processor Registers
  - Caches
  - Memory Addressing
- How processor executes HLLs?
- High-Level code <-> Assembly code <-> Machine code

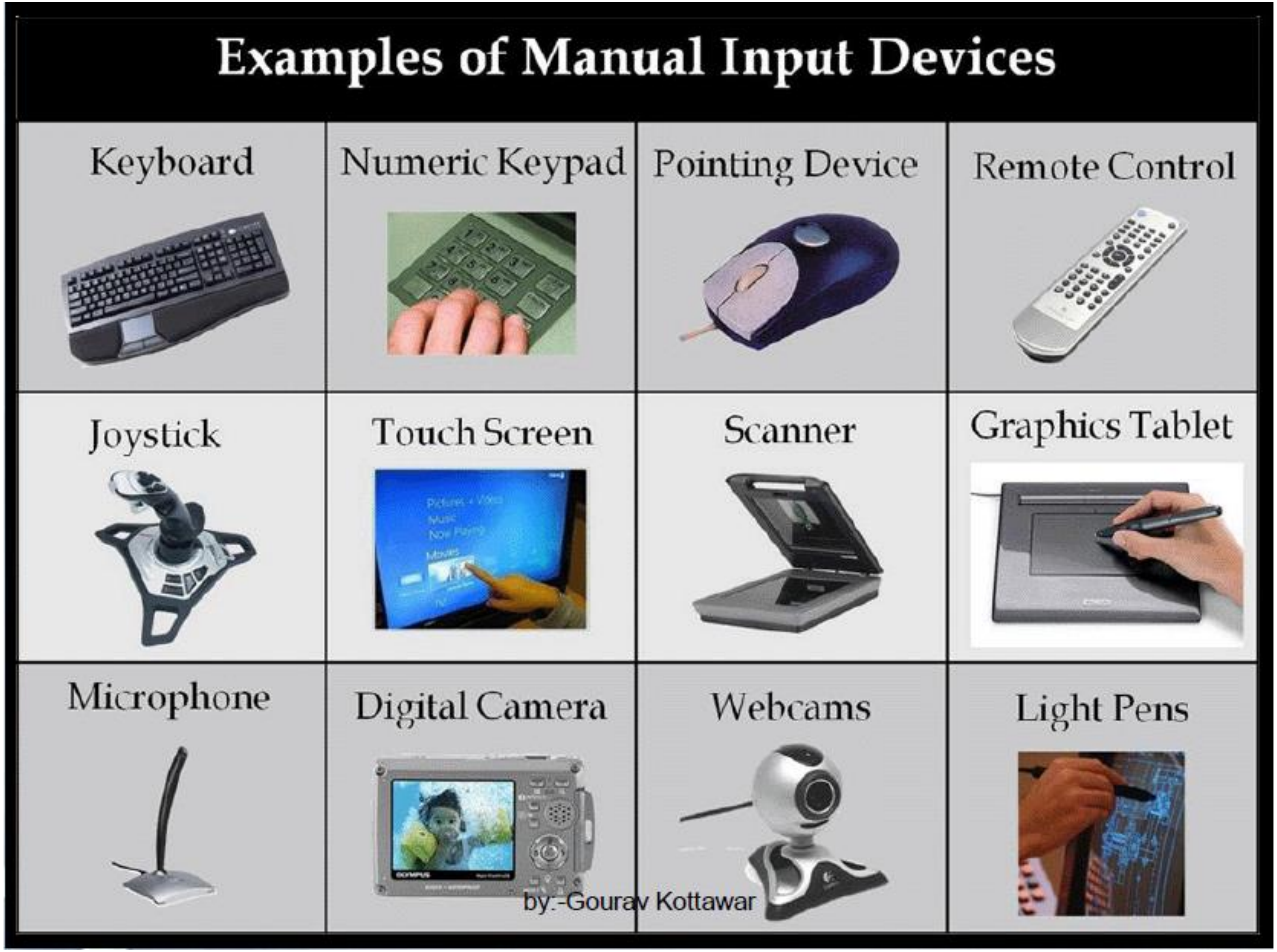
CPU : The Heart of Computing System



ca 1980
It took 10 of these boards to make a *Central Processing Unit*



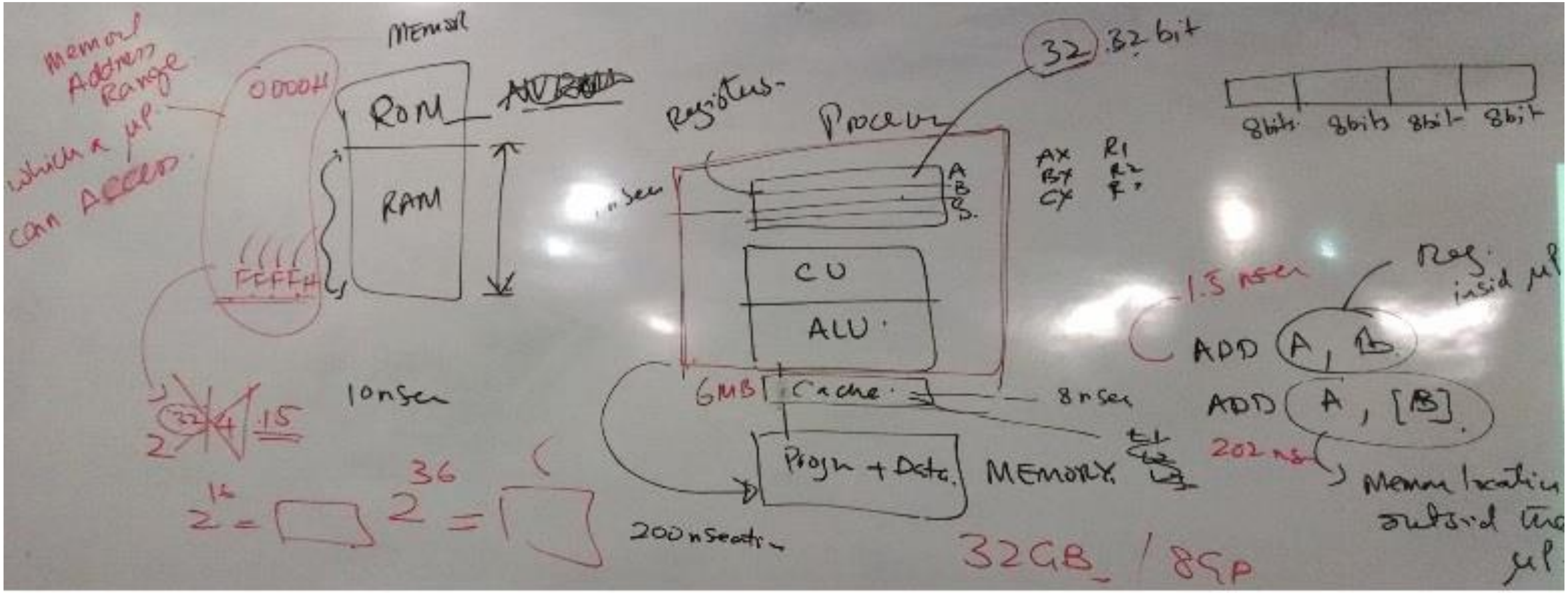
ca 2000
You can see why they called this CPU a *microprocessor*!


Examples of Manual Input Devices
Keyboard
Numeric Keypad
Pointing Device
Remote Control
Joystick
Touch Screen
Scanner
Graphics Tablet
Microphone
Digital Camera
Webcams
Light Pens
by:-Gourav Kottawar

Storage Devices

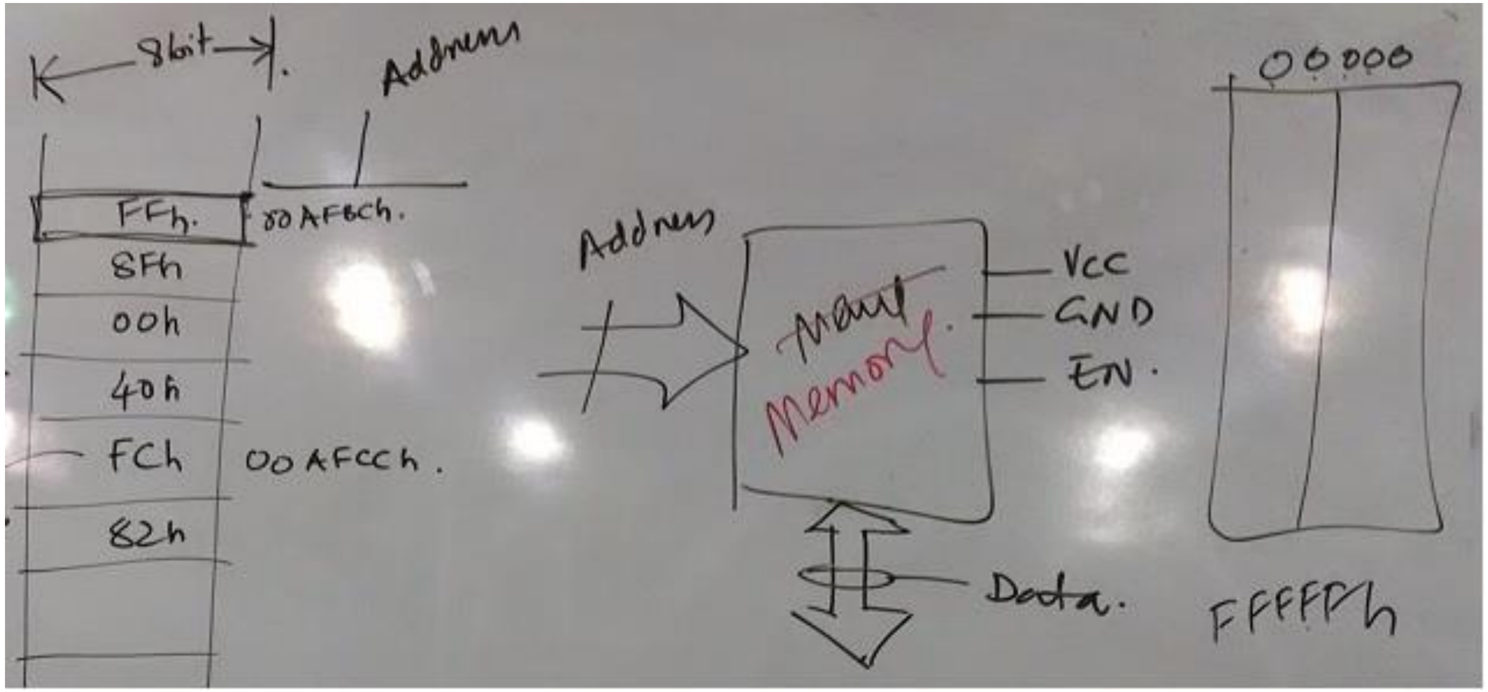
Storage Devices

Lecture # 3 Whiteboard Snaps



ROM, External Memory (RAM), Size of memory addressable by a processor, Processor Registers, Cache, ADD instruction with register operands, ADD instructions with memory operands.

Lecture # 3 Whiteboard Snaps



Lecture # 3 Whiteboard Snaps

BINARY	POWER OF 2	DECIMAL
1	$=2^0=$	1
10	$=2^1=$	2
100	$=2^2=$	4
1000	$=2^3=$	8
10000	$=2^4=$	16
100000	$=2^5=$	32
1000000	$=2^6=$	64
10000000	$=2^7=$	128
100000000	$=2^8=$	256
1000000000	$=2^9=$	512
10000000000	$=2^{10}=$	1024
100000000000	$=2^{11}=$	2048
1000000000000	$=2^{12}=$	4096
10000000000000	$=2^{13}=$	8192
100000000000000	$=2^{14}=$	16384
1000000000000000	$=2^{15}=$	32768
10000000000000000	$=2^{16}=$	65536
100000000000000000	$=2^{17}=$	131072
1000000000000000000	$=2^{18}=$	262144
10000000000000000000	$=2^{19}=$	524288
100000000000000000000	$=2^{20}=$	1048576

Hexadecimal As Shorthand for Binary

Lecture # 3 Whiteboard Snaps

HEXADECIMAL NUMERALS	PRONUNCIATION (FOLLOW WITH "HEX")	DECIMAL EQUIVALENT
0	Zero	0
1	One	1
2	Two	2
3	Three	3
4	Four	4
5	Five	5
6	Six	6
7	Seven	7
8	Eight	8
9	Nine	9
A	A	10
B	B	11
C	C	12
D	D	13
E	E	14
F	F	15
10	Ten (or, One-oh)	16
11	One-one	17
12	One-two	18

HEXADECIMAL NUMERALS	PRONUNCIATION (FOLLOW WITH "HEX")	DECIMAL EQUIVALENT
13	One-three	19
14	One-four	20
15	One-five	21
16	One-six	22
17	One-seven	23
18	One-eight	24
19	One-nine	25
1A	One-A	26
1B	One-B	27
1C	One-C	28
1D	One-D	29
1E	One-E	30
1F	One-F	31
20	Twenty (or, Two-oh)	32

```
  1       1
  2 F 3 1 A DH
+ 9 6 B A 0 7H
--------------
  C 5 E B B 4H
```

Hexadecimal is the programmer's shorthand for the computer's binary numbers.

```
1111 0000 0000 0000 1111 1010 0110 1110
 F    0    0    0    F    A    6    E
```

How to use Processor to execute High-level languages?

- We program in High-level languages.
- Processor = Micro-architecture + ISA
- In order to execute program on a particular processor, we need to covert HLL code into processor specific machine code (obeying the ISA).
- How to fill this gap? High-level code vs machine-code
  - Compilers fill this gap by reading high-level language programs and generating low-level language (machine-code) for execution on a processor.
  - Write in machine-code OR manually convert high-level code into machine-code OR writing in some short hand language. (extreme way to understand processor design)
- Processors have extra digital circuitry which does not execute code but provide support for (or speed up) execution.

High-Level code <-> Assembly code <-> Machine code

```c
// Type your code here, or load an example.
#include <stdio.h>

int square(int num);

int main (void) {
    int v_num = 10, v_res = 0;
    v_res = square (v_num);
    printf("Square is %d \n");
}

int square(int num) {
    return num * num;
}
```

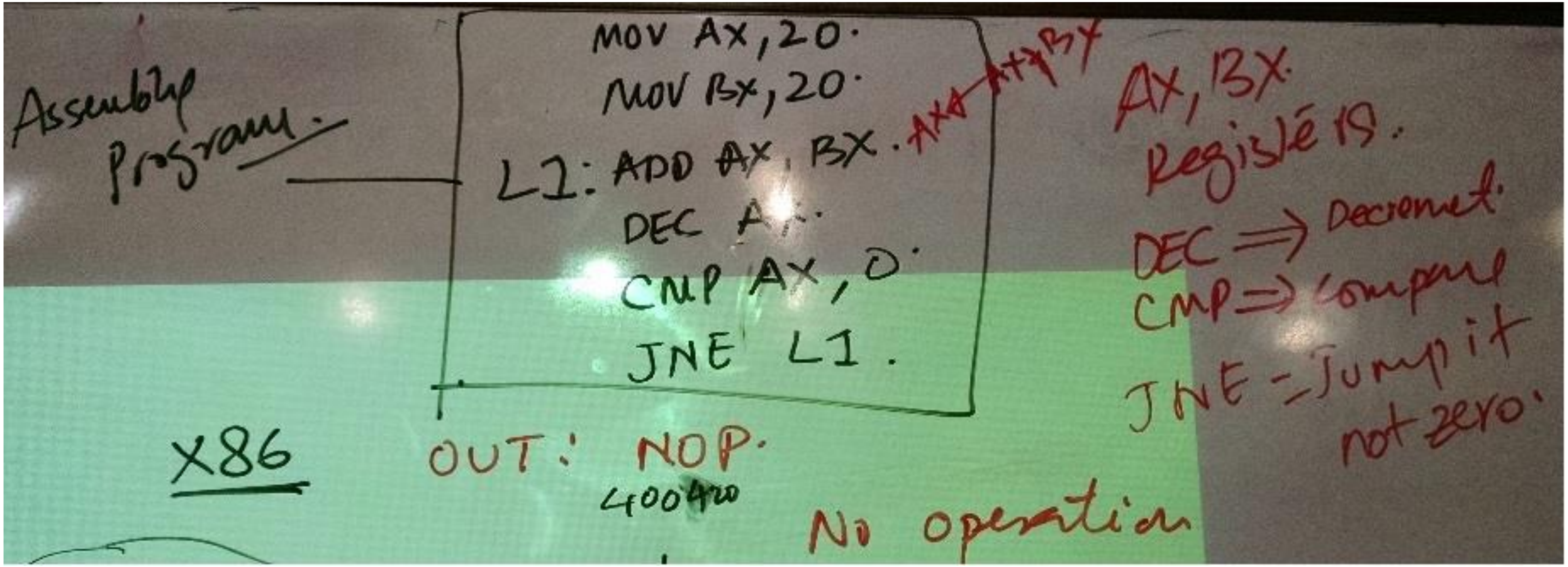
```asm
.LC0:
        .string "Square is %d \n"
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     DWORD PTR [rbp-4], 10
        mov     DWORD PTR [rbp-8], 0
        mov     eax, DWORD PTR [rbp-4]
        mov     edi, eax
        call    square(int)
        mov     DWORD PTR [rbp-8], eax
        mov     edi, OFFSET FLAT:.LC0
        mov     eax, 0
        call    printf
        mov     eax, 0
        leave
        ret
square(int):
        push    rbp
        mov     rbp, rsp
        mov     DWORD PTR [rbp-4], edi
        mov     eax, DWORD PTR [rbp-4]
        imul    eax, DWORD PTR [rbp-4]
        pop     rbp
        ret
```

```
400420  ff 25 f2 0b 20 00
400426  68 00 00 00 00
40042b  e9 e0 ff ff ff

400460  f3 c3
400462  66 2e 0f 1f 84 00 0
40046c  0f 1f 40 00

400512  55
400513  48 89 e5
400516  48 83 ec 10
40051a  c7 45 fc 0a 00 00 0
400521  c7 45 f8 00 00 00 0
400528  8b 45 fc
40052b  89 c7
40052d  e8 19 00 00 00
400532  89 45 f8
400535  bf e4 05 40 00
40053a  b8 00 00 00 00
40053f  e8 dc fe ff ff
400544  b8 00 00 00 00
400549  c9
40054a  c3
40054b  55
40054c  48 89 e5
40054f  89 7d fc
400552  8b 45 fc
400555  0f af 45 fc
400559  5d
40055a  c3
40055b  0f 1f 44 00 00
```

Home work (Sec A)



Home work (Sec E)