Lecture - 11

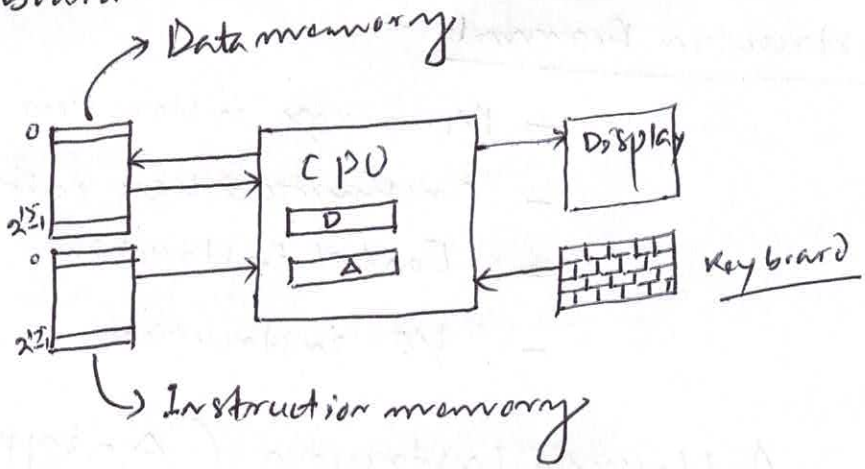
Last lecture

- fundamental concept
  - Instruction format
  - Addressing mode
  - RISC type (fixed format)
  - CISC type (variable format)
  - Semi variable format
- Example
  - MIPS
  - x86

Hack computer specification :-

Main component of Hack -

One
- ACPU (ALU + control unit)
- Two registers
- Two memory
- Display screen
- Keyboard

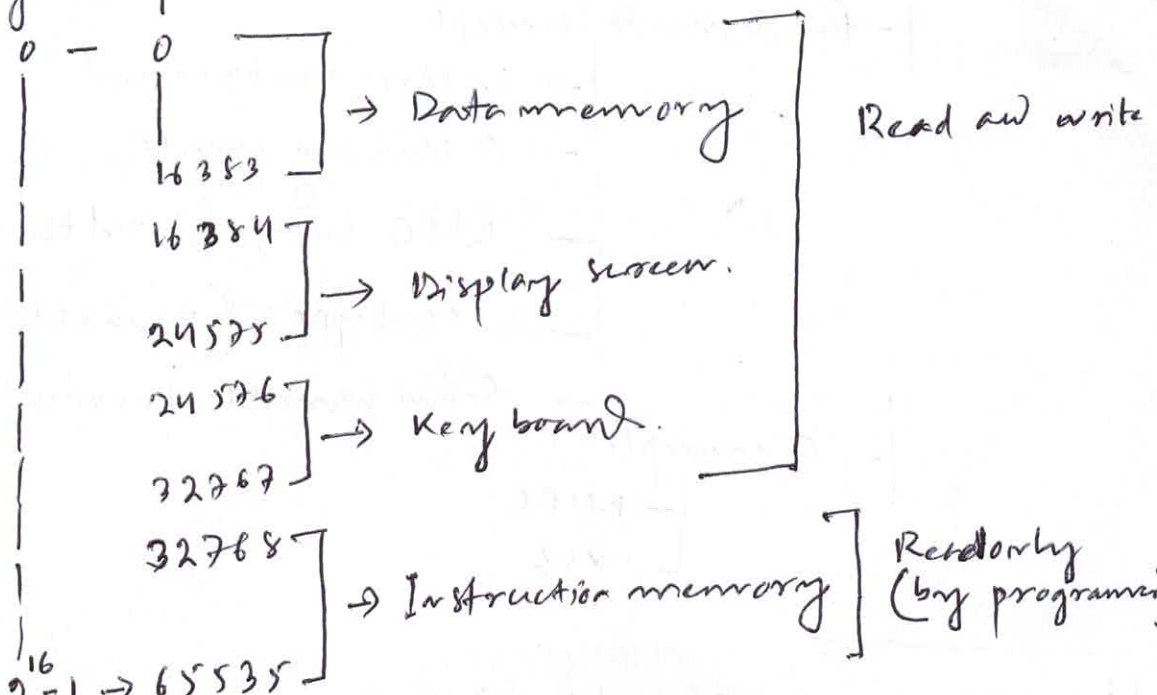


Address space :-

- 16-bit memory address space
- Address range : 0 to $2^{16}-1$

(2)

- Memory maps:



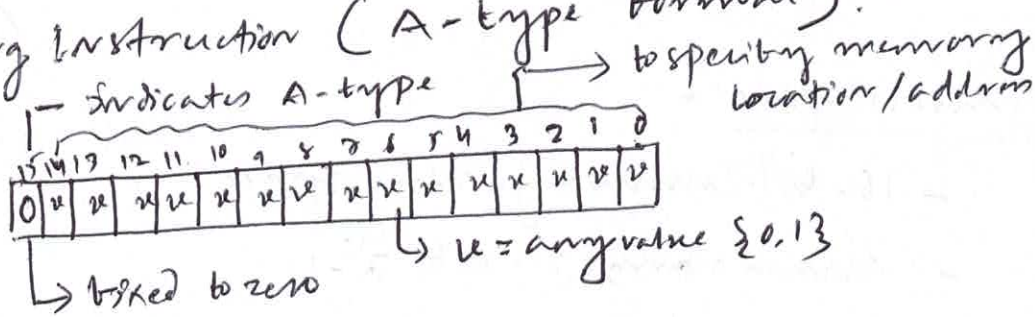
Addressing mode principle:-

- Dedicated data register 'D'
- Dedicated address register 'A', also can be used for data.
- All the memory related instruction ~~implicitely~~ access the implicit address specified via register A.

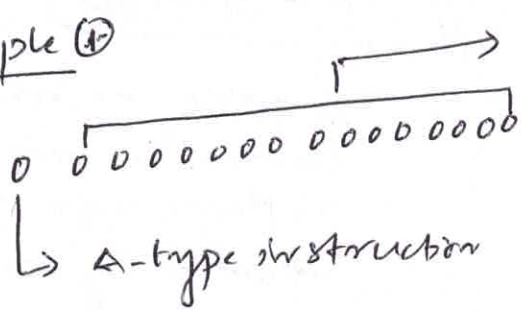
Instruction Format:-

- Memory instructions
- Arithmetic & logic instructions
- Control instructions
- I/O instructions

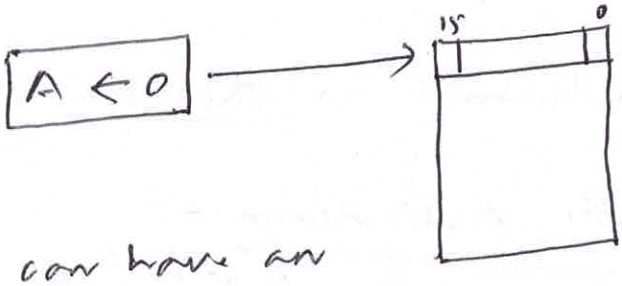
Addressing Instruction (A-type format):-



Example (1)



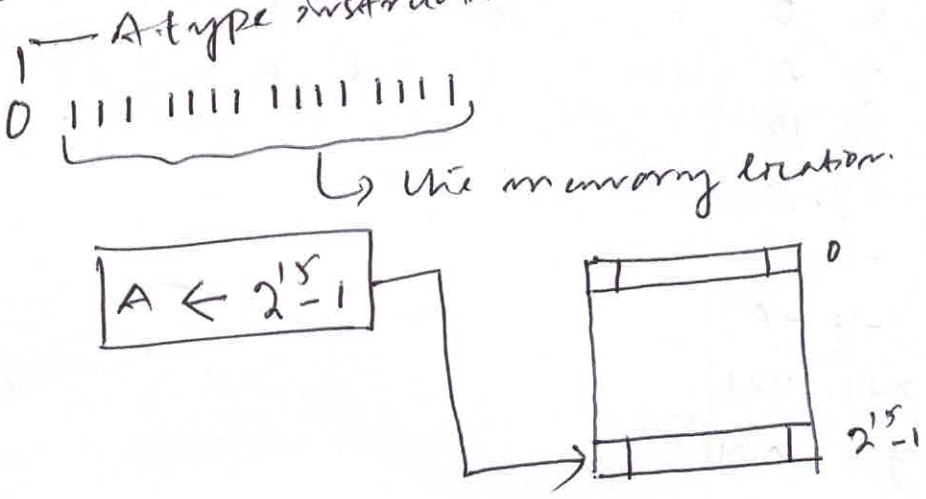
meaning :-



The register A can have an unsigned integer value in the range of

0 to $2^{15}-1$

Example (2) :-



Assembly code for A-type instruction :-

@ value // value could be any non negative number in range 0 to $2^{15}-1$.

Example:

① @0 // meaning :- A ← 0

② @15 // A ← 15

③ @32767 // A ← 32767

④ @symbol // A ← symbol

(11)

@variable // A ← variable

variable is a memory location. The name "variable" is replaced with some memory location by assembler.

Instruction format for ALU and Control instruction :-

Recall the ALU design :-

			Register A as data register						Register A as address register
			c_1	c_2	c_3	c_4	c_5	c_6	
	a=0		2x	nx	2y	ny	+	ND	a=1
0	0	0	1	0	1	0	1	0	
1	1	1	1	1	1	1	1	1	
2	-1	-1	1	1	1	0	1	0	
3	x	D	0	0	1	1	0	0	
4	y	A	1	1	0	0	0	0	M 18
5	!x	!D							!M 19
6	!y	!A							
7	-x	-D							-M 20
8	-y	-A							
9	x+1	D+1							M+1 21
10	y+1	A+1							
11	x-1	D-1							M-1 22
12	y-1	A-1							
13	x+y	D+A							D+M 23
14	x-y	D-A							D-M 24
15	y-x	A-D							M-D 25
16	x&y	D&A							D&M 26
18	x|y	D|A	0	1	0	1	0	1	D|M 27

x is mapped with register D

y is mapped with register A

Specification for destination operand :-

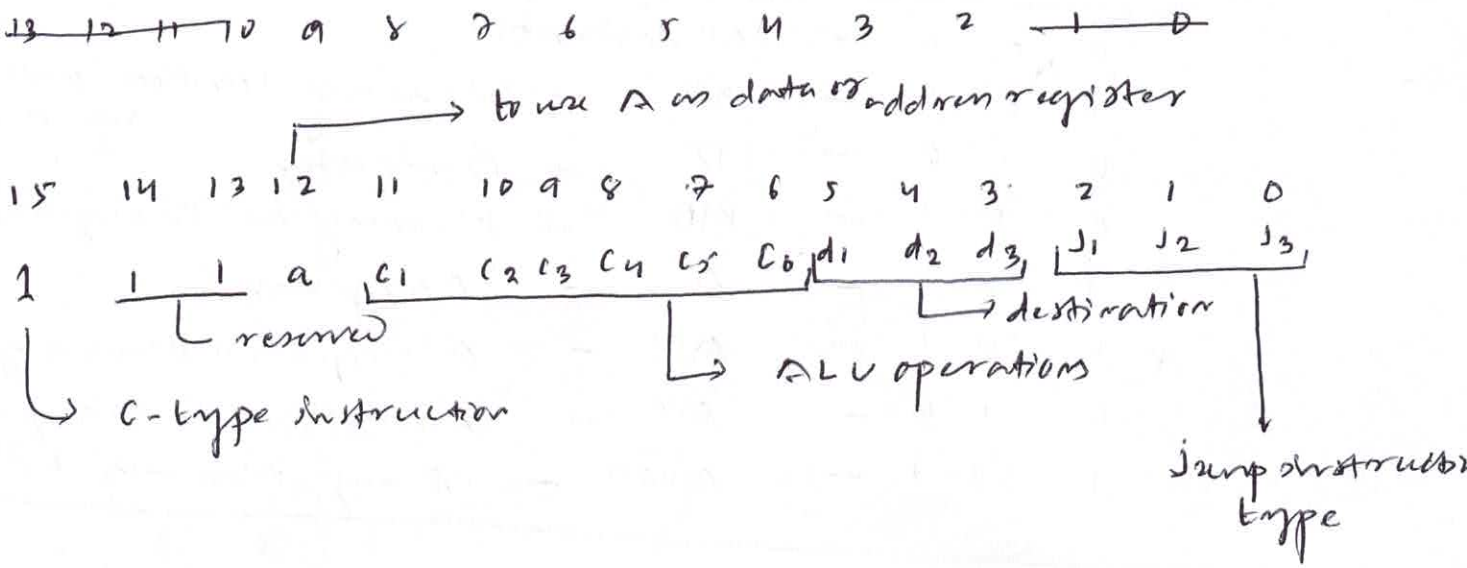
	d_1	d_2	d_3	Mnemonic & Meaning
	0	0	0	— no destination
	0	0	1	— M — Memory location addressed by A
	0	1	0	— D — D register
	0	1	1	— MD — Memory and D register
	1	0	0	— A — A register
*	1	0	1	— AM — A register & Memory
	1	1	0	— AD — A register and D reg.
	1	1	1	— AMD — A reg, Memory & Dre

Specification for Jump instruction (Conditional & unconditional)

Effect	J_1	J_2	J_3	Mnemonic / Meaning
No jump	0	0	0	nothing
jp out > 0 : jump	0	0	1	JGT
jp out = 0 : jump	0	1	0	JEQ
jp out ≥ 0 : jump	0	1	1	JGE
jp out < 0 : jump	0	0	0	JLT
jp out ≠ 0 jump	1	0	1	JNE
jp out ≤ 0 jump	1	1	0	JLE
Jump	1	1	1	JMP

(6)

The C-type instruction format (ALU & control) :-



Assembly language mnemonic :-

alu related instruction :-

```
dest = comp
```

Example :- ① M = D+M

// Memory[A] ← D + Memory[A]

② D = D - A

// D ← D - A

Jump instruction :-

```
cond jump
```

if cond- condition then jump

Example :- ① D ; JEQ

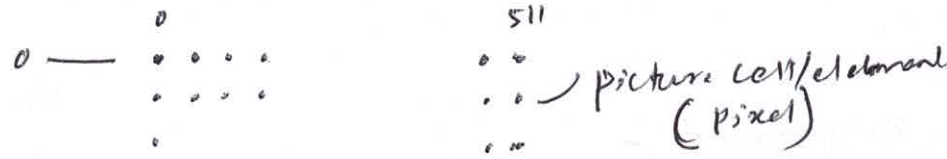
// if D=0 then jump to location pointed by 'A'

Dealing with I/O devices :-

- Display screen
- Keyboard

Display screen – Black (1) and white (0)

Dimension.



Memory Required :-

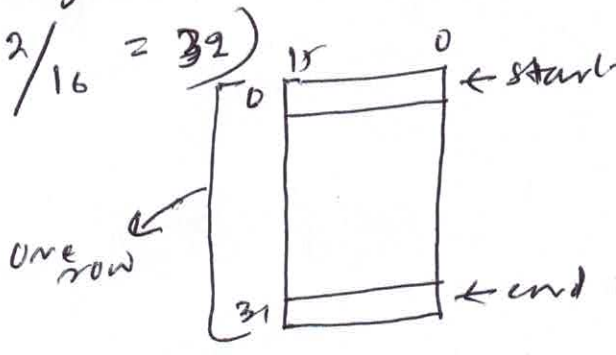
$$2^8 \times 2^9 = 2^{10} \times 2^3 \times 2^4$$

$$= 8K \text{ word addressable memory}$$

To store one row of display screen we need.

32 consecutive memory location

$\left(512/16 = 32 \right)$



256 set of such locations are needed to store the complete screen.

(8)

Example instructions :-

To display black dots in first 16 pixel

@SCREEN // setting the register A with the location of display.

M = -1 // -1 = 1111 1111 1111 1111

Keyboard :-

- A single word memory location is used
- mapped to location 24576