

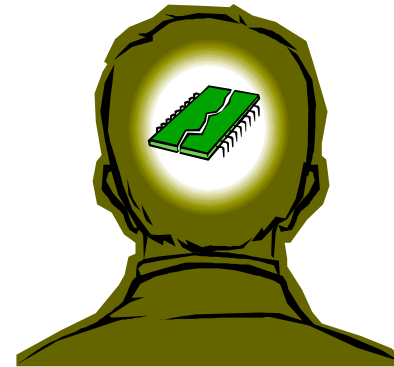
# Memories in C-Language



Hoon Yoo, Ph.D.

# Handling Memory

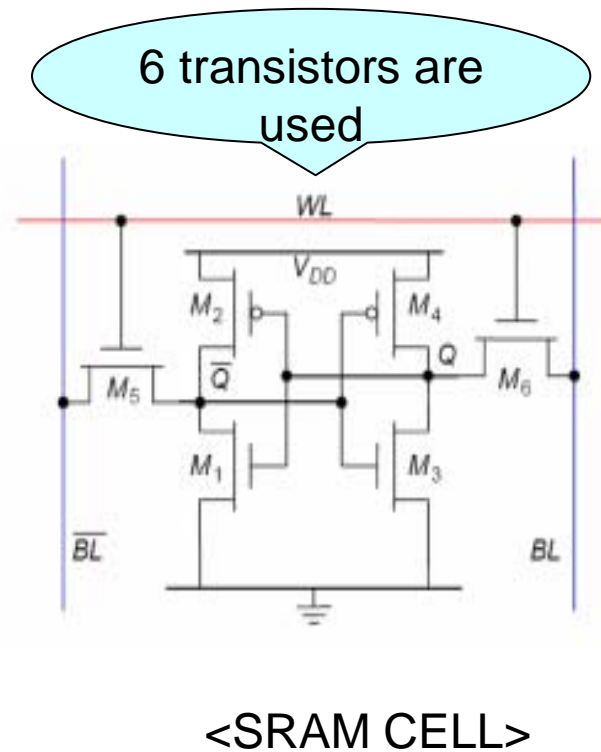
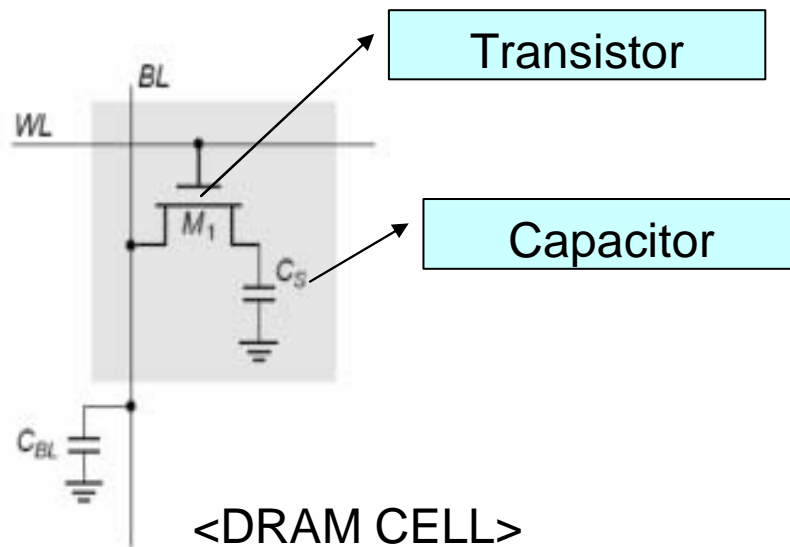
- Hardware memory
  - Volatile memory
    - DRAM (dynamic RAM)
      - SDRAM (synchronous DRAM)
    - SRAM (static RAM)



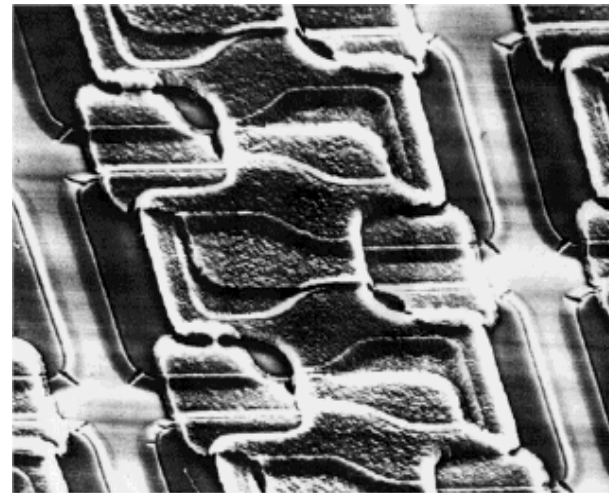
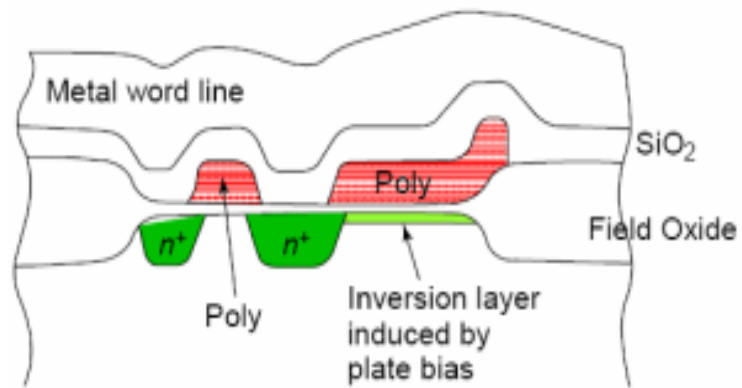
	DRAM	SRAM
Speed	Slow	Fast
Cost	Low	High
Refresh	Required	-

# Handling Memory

- One bit memory



# Handling Memory



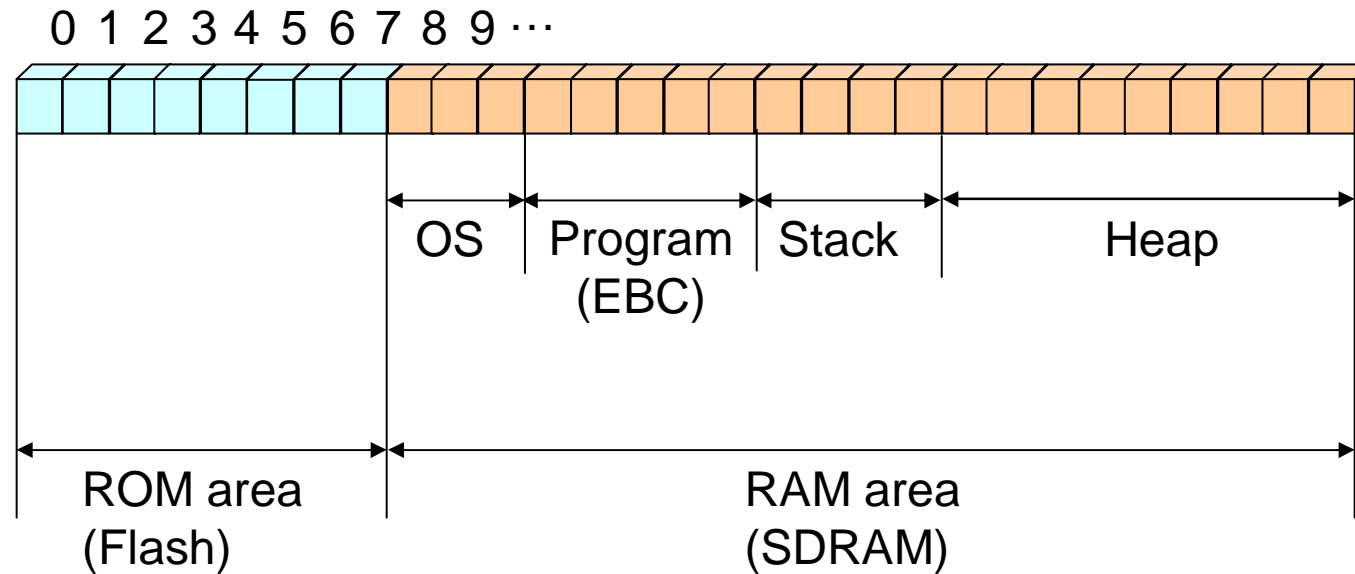
# Handling Memory

- Hardware memory
  - Non-volatile memory
    - ROM
    - PROM (Programmable ROM)
    - EPROM (Erasable PROM)
    - EEPROM (Electrical EPROM)
    - Flash (Flash EEPROM)
      - NAND flash
      - NOR flash
    - Optical (CD/DVD)



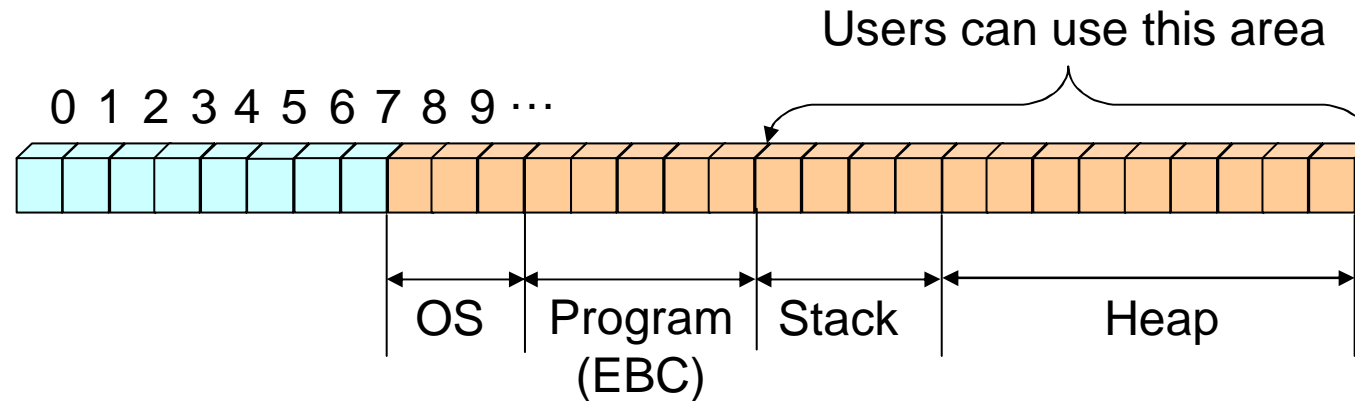
# Handling Memory

- Memory in software



# Handling Memory

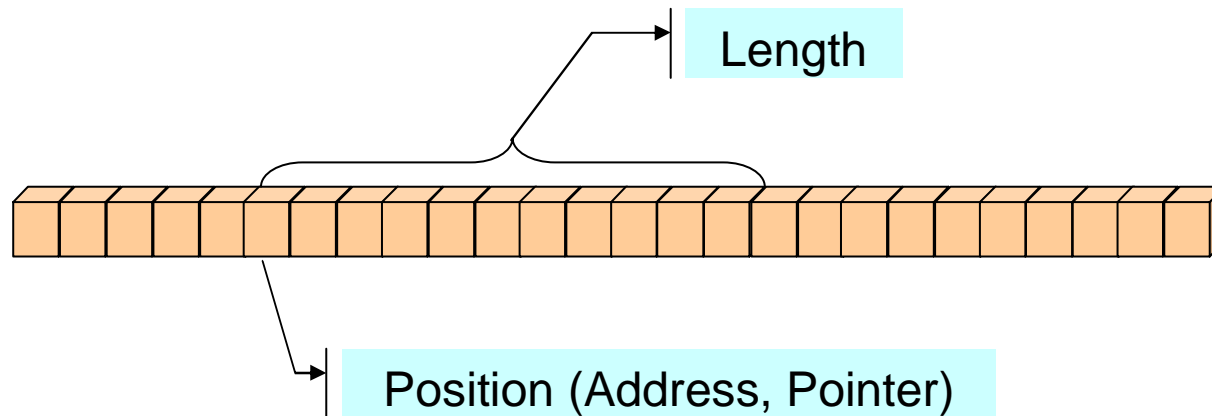
- Memory in software



	Stack	Heap
How to create	char, int, float, etc	malloc(), etc
Who allocates	automatic	User
Who frees	automatic	User

# Handling Memory

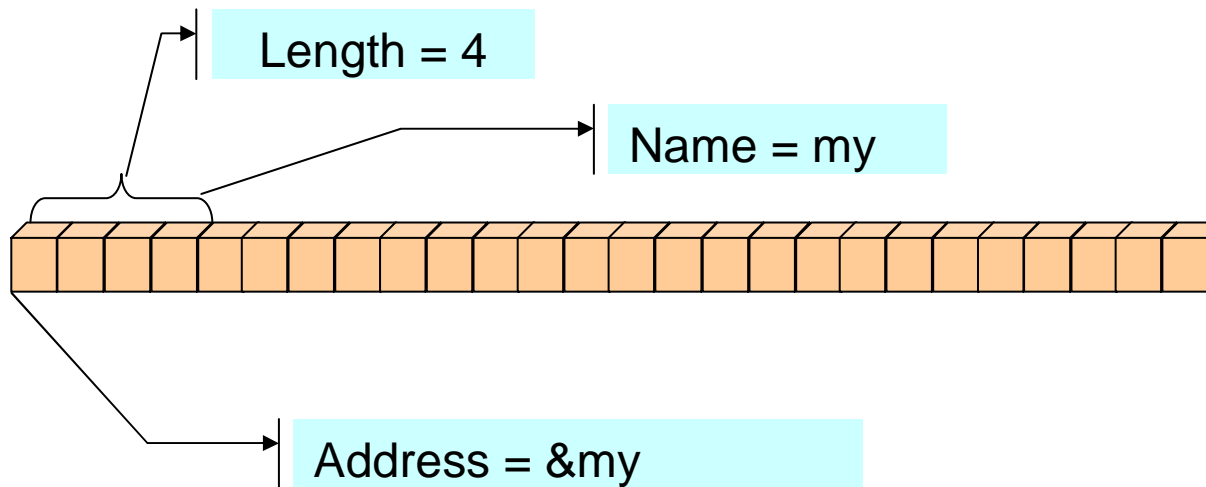
- Parameters for memory in software
  - Starting position (Address, Pointer)
  - Length
  - Type
    - Integer – unsigned or signed
    - Real – double, float





# Handling Memory

```
main()
{
    unsigned int my = 5;
}
```



# Handling Memory

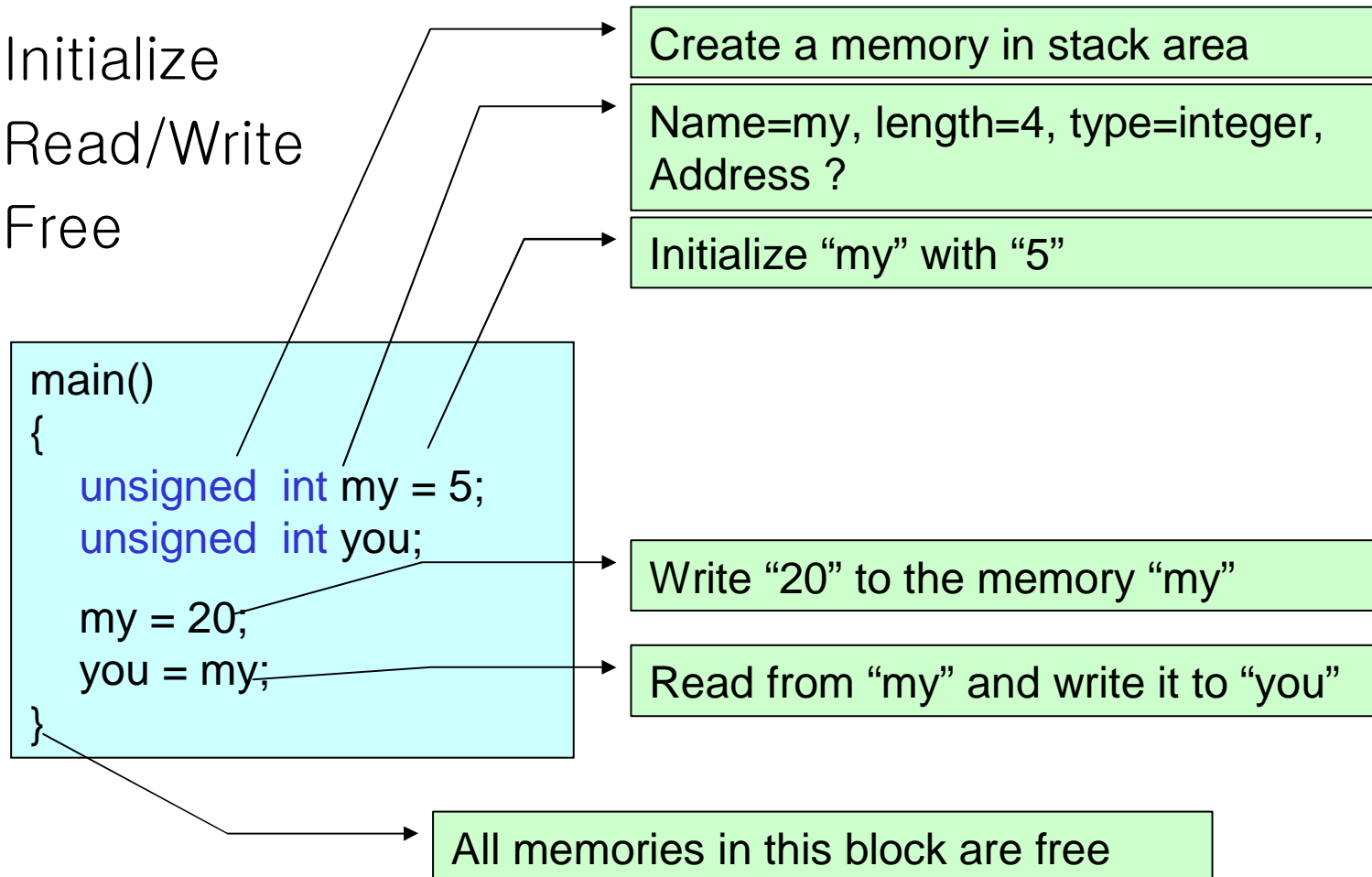
- Memory Management

- Creation

- Initialize

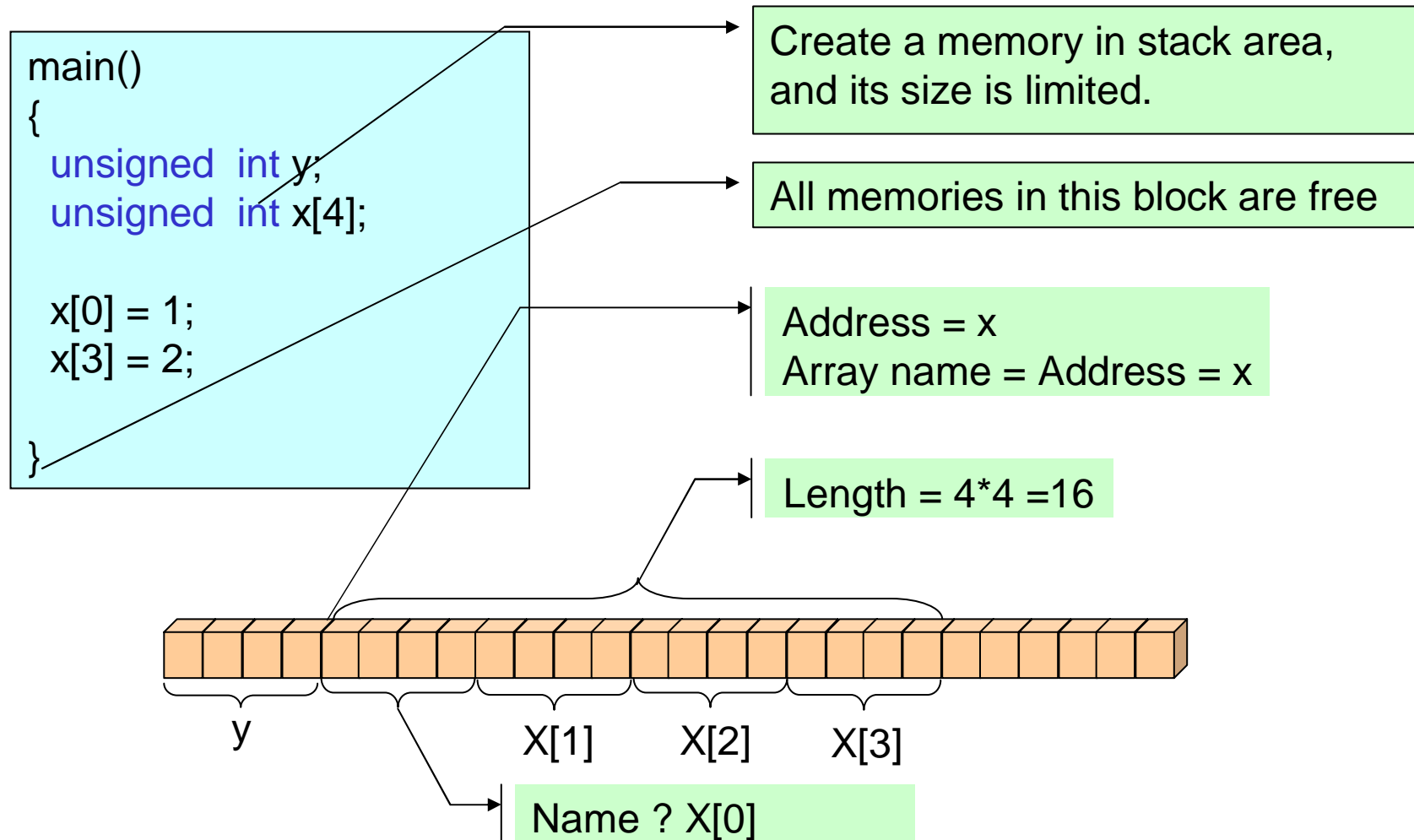
- Read/Write

- Free



# Handling Memory

- How to create mass memories



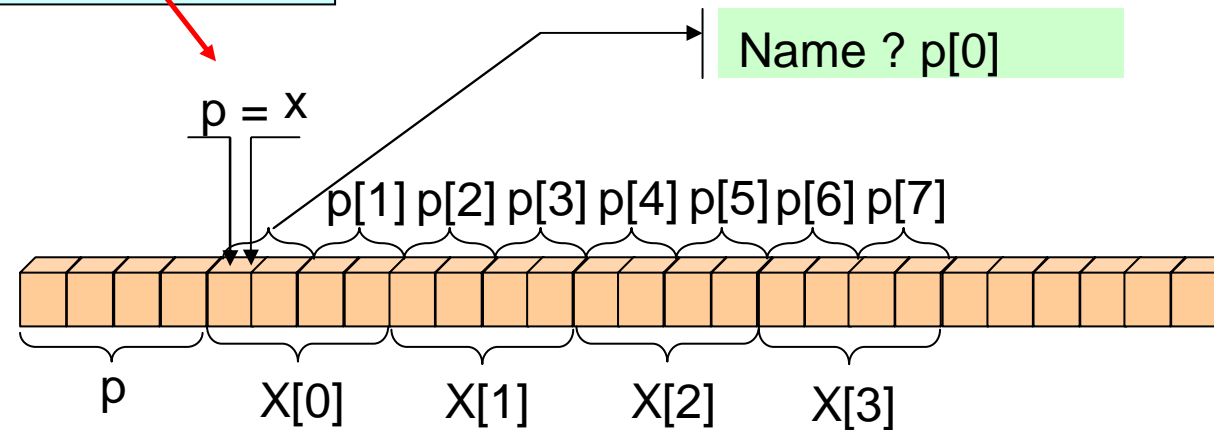
# Handling Memory

- How to create mass memories in the stack

```
main()
{
    unsigned short *p;
    unsigned int x[4];

    p = (short*)x;
    p[2]=1;
}
```

The memory type  
of left and right  
should be same in  
the operator “=”.  
If not, you need the  
cast operation.



# Handling Memory

- How to create mass memories in the global area

```
unsigned char buf[1024];  
main()  
{  
    unsigned short *p;  
  
    p = (short*)buf;  
    p[2]=1;  
}
```

- Declare a global memory
  - ➔ Allocate it when program starts
  - ➔ De-allocate it when program ends
- Large memories can be allocated.

- buf[0] ~ buf[1023]
- p[0] ~ p[??]

# Handling Memory

- How to create mass memories in the heap

```
main()
{
    unsigned char *buf;
    unsigned short *p;

    buf = (unsigned char *)malloc(1024);
    memset(buf, 0, 1024);

    p = (short*)buf;
    p[2]=1;
    ...

    free(buf);
}
```

• Prepare a pointer (address variable)

• Allocate a memory in the heap

• Initialize the memory

• De-allocate the memory in the heap

# Summary

• Key parameters for
Address
Length

• Handling memory
Prepare pointers (address)
Allocate
Initialize
Read/Write
De-allocate