CS & IT ENGINEERING

Operating Systems

Memory Management

Lecture No. 1

By- Dr. Khaleel Khan Sir

TOPICS TO BE COVERED

- Abstract View of Memory
- Loading vs Linking
- Address Binding

Q.8 Consider the following snapshot of a system running n processes. Process i is holding xi instances of a resource R, for $1 \leq i \leq n$. Currently, all instances of R are occupied. Further, for all i, process i has placed a request for an additional yi instances while holding the xi instances it already has. There are exactly two processes p and q such that $yp = yq = 0$. Which one of the following can serve as a necessary condition to guarantee that the system is not approaching a deadlock?

A. $\min(x_p, x_q) < \max_{k \neq p, q} y_k$

B. $x_p + x_q \geq \min_{k \neq p, q} y_k$

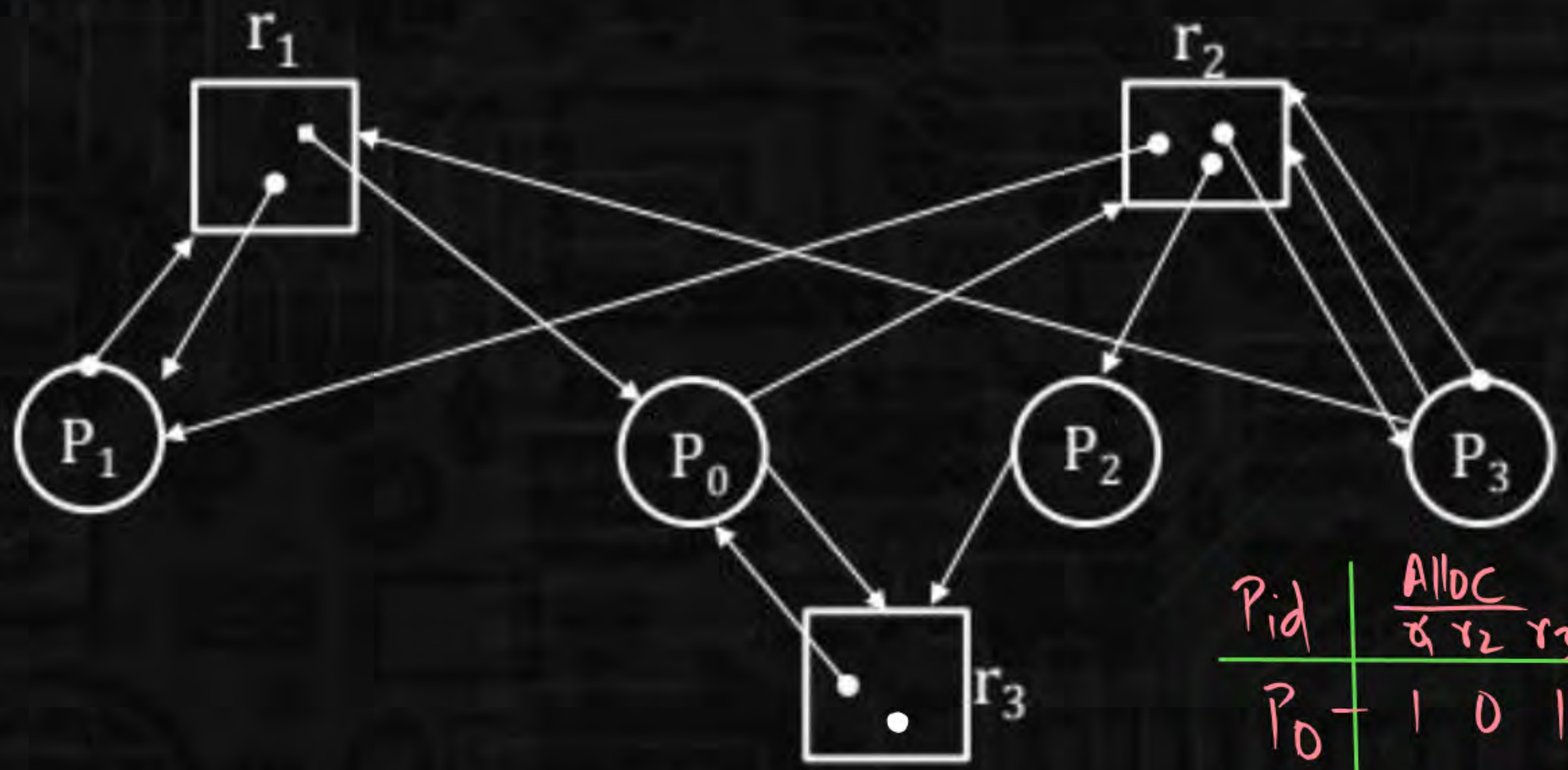
C. $\max(x_p, x_q) > 1$

D. $\min(x_p, x_q) > 1$

Pid	Alloc R	Req R	Avail R
P_1	x_1	y_1	0
P_2	x_2	y_2	
P_3	x_3	y_3	
⋮	⋮		
P_p	x_p	0	
P_q	x_q	0	
⋮	⋮		
P_n	x_n	y_n	

$\underline{\underline{I}}: (x_p + x_q) \geq \min_{k \neq (p,q)} (y_k)$

Q.9 Consider the Following Resource Allocation Graph. Find if the System is in Deadlock State.



Safe ✓ $\langle P_2; P_0; P_1; P_3 \rangle$
vs
deadlock

Pid	Alloc r_1	Alloc r_2	Alloc r_3	Req r_1	Req r_2	Req r_3	Avail r_1	Avail r_2	Avail r_3
P_0	1	0	1	0	1	1	0	0	1
P_1	1	1	0	1	0	0			
P_2	0	1	0	0	0	1			
P_3	0	1	0	1	2	0			

Q.10 An operating system uses the *Banker's algorithm* for deadlock avoidance when managing the allocation of three resource types X, Y, and Z to three processes P0, P1, and P2. The table given below presents the current system state. Here, the *Allocation* matrix shows the current number of resources of each type allocated to each process and the *Max* matrix shows the maximum number of resources of each type required by each process during its execution.

	Allocation			Max		
	X	Y	Z	X	Y	Z
P0	0	0	1	8	4	3
P1	3	2	0	6	2	0
P2	2	1	1	3	3	3

There are 3 units of type X, 2 units of type Y and 2 units of type Z still available. The system is currently in a safe state. Consider the following independent requests for additional resources in the current state:

Q. [redacted]

REQ1: P0 requests 0 units of X, 0 units of Y and 2 units of Z
REQ2: P1 requests 2 units of X, 0 units of Y and 0 units of Z

Which one of the following is TRUE?

A. Only REQ1 can be permitted.

B. Only REQ2 can be permitted.

C. Both REQ1 and REQ2 can be permitted.

D. Neither REQ1 nor REQ2 can be permitted.

Q.13 A system contains three programs, and each requires three tape units for its operation. The minimum number of tape units which the system must have such that deadlocks never arise is ___7___.

	Need	Alloc
P_1	3	2
P_2	3	2
P_3	3	2

$+1 = 7$

Q.14 A system shares 9 tape drives. The current allocation and maximum requirement of tape drives for three processes are shown below:

Which of the following best describes current state of the system?

Process	Current Allocation	Maximum Requirement
P1	3	7
P2	1	6
P3	3	5

Need
4
5
2

Avail
~~2~~ 5

A. Safe, Deadlocked

B. Safe, Not Deadlocked

C. Not Safe, Deadlocked

D. Not Safe, Not Deadlocked

$\langle P_3; P_1; P_2 \rangle$

Q.15 Which of the following statements is/are TRUE with respect to deadlocks?

MSQ

<A, D>

A. Circular wait is a necessary condition for the formation of deadlock.

B. In a system where each resource has more than one instance, a cycle in its wait-for graph indicates the presence of a deadlock.

C. If the current allocation of resources to processes leads the system to unsafe state, then deadlock will necessarily occur.

D. In the resource-allocation graph of a system, if every edge is an assignment edge, then the system is not in deadlock state.

Q.18 A multithreaded program P executes with x number of ~~threads~~ Processes and uses y number of locks for ensuring mutual exclusion while operating on shared memory locations. All locks in the program are non-reentrant, i.e., if a thread holds a lock l, then it cannot re-acquire lock l without releasing it. If a thread is unable to acquire a lock, it blocks until the lock becomes available. The minimum value of x and the minimum value of y together for which execution of P can result in a deadlock are:

A. $x = 1, y = 2$

B. $x = 2, y = 1$

C. $x = 2, y = 2$

D. $x = 1, y = 1$

2 2

Pr_i

P(x);
P(y);

2 1

Pr_j

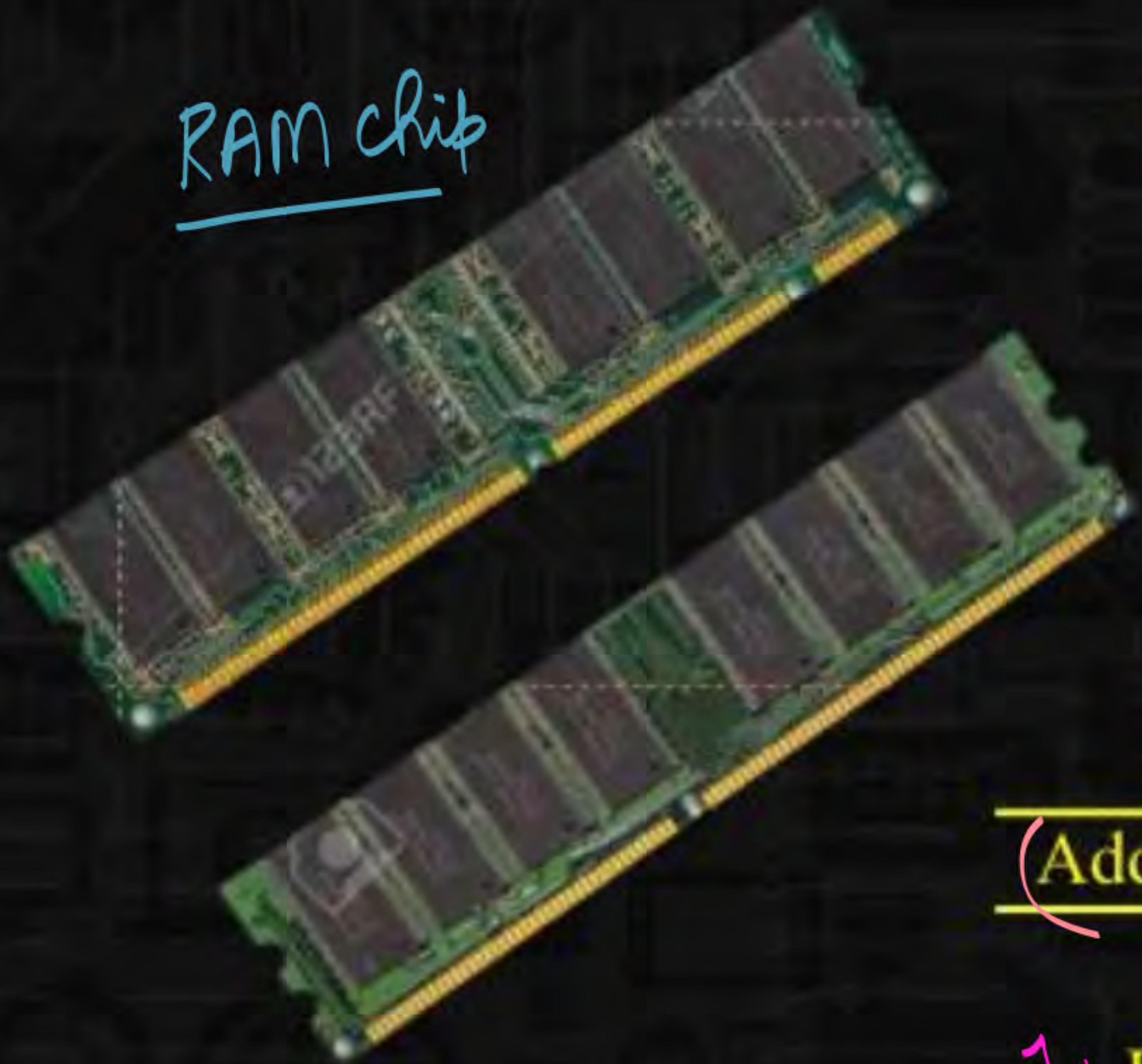
P(y);
P(x);

Pr_i

P(x);
P(x);

Main Memory Access Time (MMAT)

RAM chip

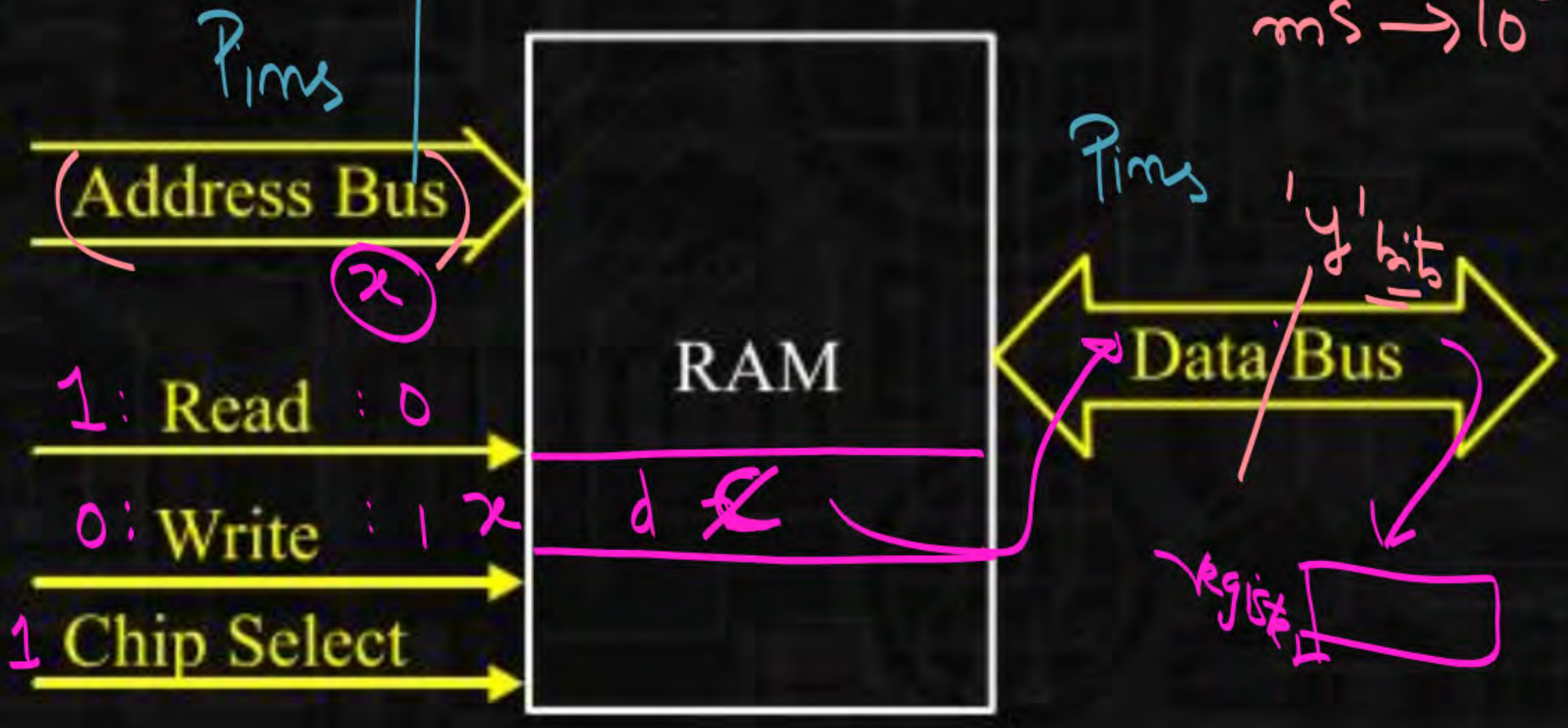


→ ns (10^{-9})

μs → 10^{-6}

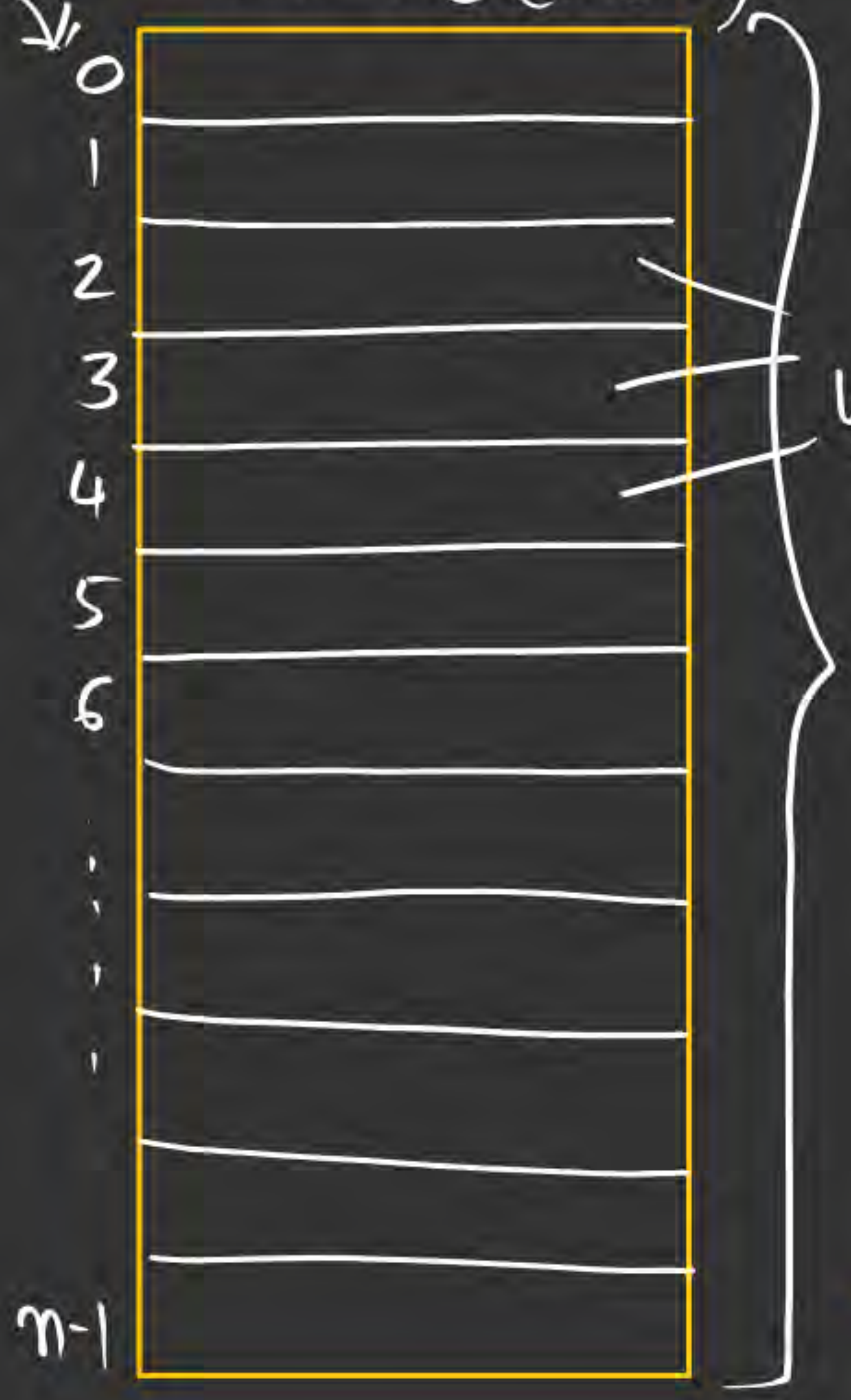
ms → 10^{-3}

group of wires (1 wire - 1 bit)



Abstract view of Memory

Memory (ROM)



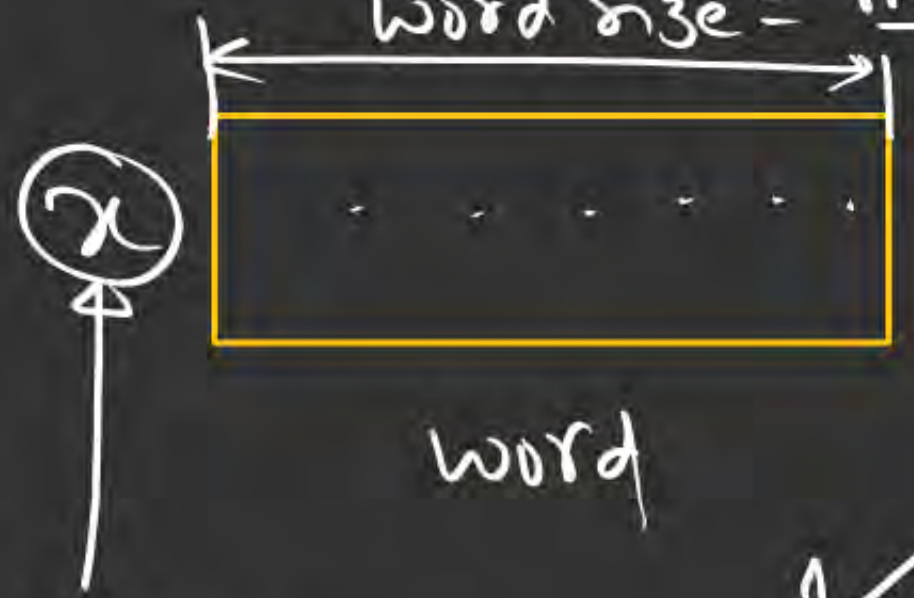
N = No. of words

Capacity

words
(Instn / data)

Bytes

word Size = 'm' bits



word

Address
(n-bits)

$1 <^{0}_{1}$

00
01
10
11

Memory Specification:

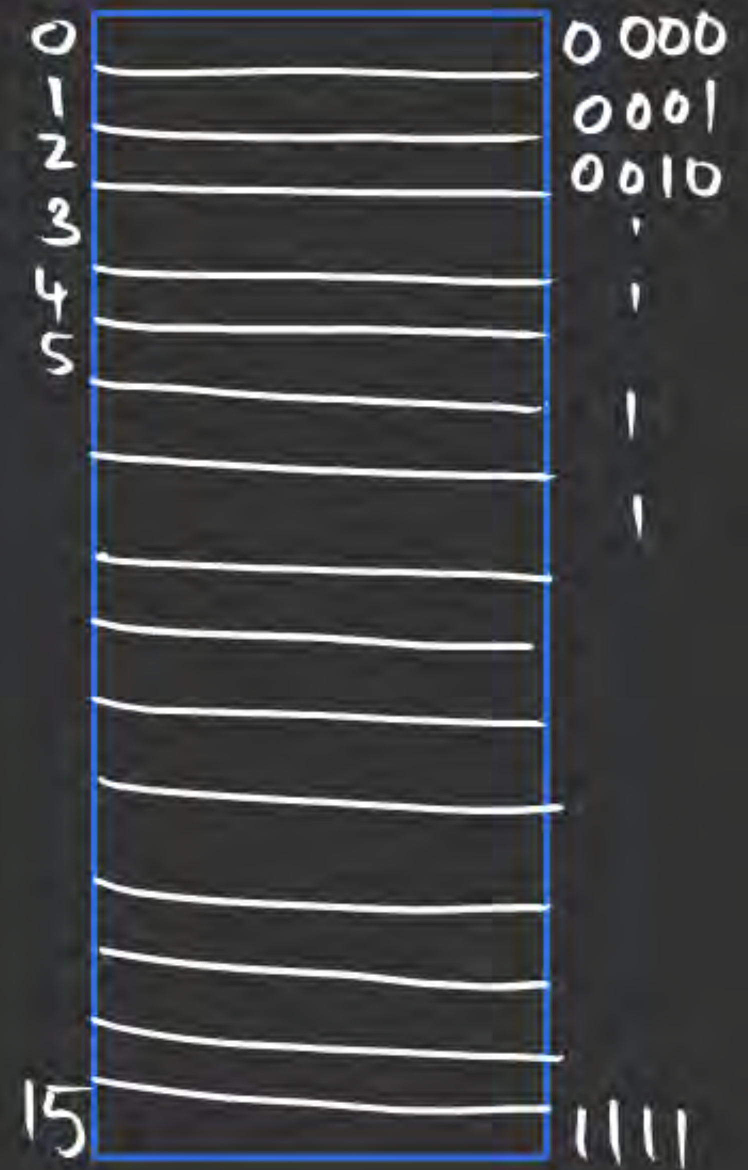
$N \times m$

I: N & n

Man

a) $N = 16\,W$

$n = 4$ bit
(Min.)



b) $N = 4GB$ $[1W = 1B]$

$n = 2 + 30 = 32$ bits

$2^{32} = 2^{2} \times 2^{30} = 4GB$ ✓

---

c) $N = 1000\,MB$

$n = 10 + 20 = 30$ bits

---

d) $n = 24$ bits

$N = 2^{24} = 2^{4} \times 2^{20}$

$= 16MW$ $(1W = 1B)$

$= 16MB$

$N = 2^{n}\,W$

$n = \log_2 N$ bits

---

e) $n = 22$ bits

$m = 16$ bits (word length)

i) $N_W = 2^{22} = 4MW$

$N_B = 4M \times 2B = 8MB$

1 Word = 2B

$4MW = ?$

Word vs Byte

$2^{5} = 32W$

$2^{6} = 64W$

$2^{7} = 128W$

$2^{8} = 256W$

$2^{9} = 512W$

$2^{10} = 1024 \sim 10^{3} = 1K$

$2^{20} = 10^{6} = 1M$

$2^{30} = 10^{9} = 1G$

$2^{40} = 10^{12} = 1T$

$2^{50} = 10^{15} = 1P$

f) $n = 34$ bit

$m = 128$ bit $(1W = 16B)$

$N_W = 2^{34} = 16GW$

$N_B = 16 \times 16 GB$
$= 256 GB$

g) $N = 512GB$

* $m = 32$ bit (word size) $(4B)$

$N_{words} = \frac{512GB}{4B} = 2^7 GW$
$= 128GW$

h) $N = (\log_2 x)$ Bytes

$n = \log \log_2 x$ bit

'N' words

$n = \log_2 N$ bit

(i) $n = \log(\log_2 c)$ bit

$N = \log_2 c \quad \left[2^{\log_2 (\log_2 c)} \right]$

$a^{\log_c b} = b^{\log_c a}$

$(\log_2 c)^{\log_2 2}$

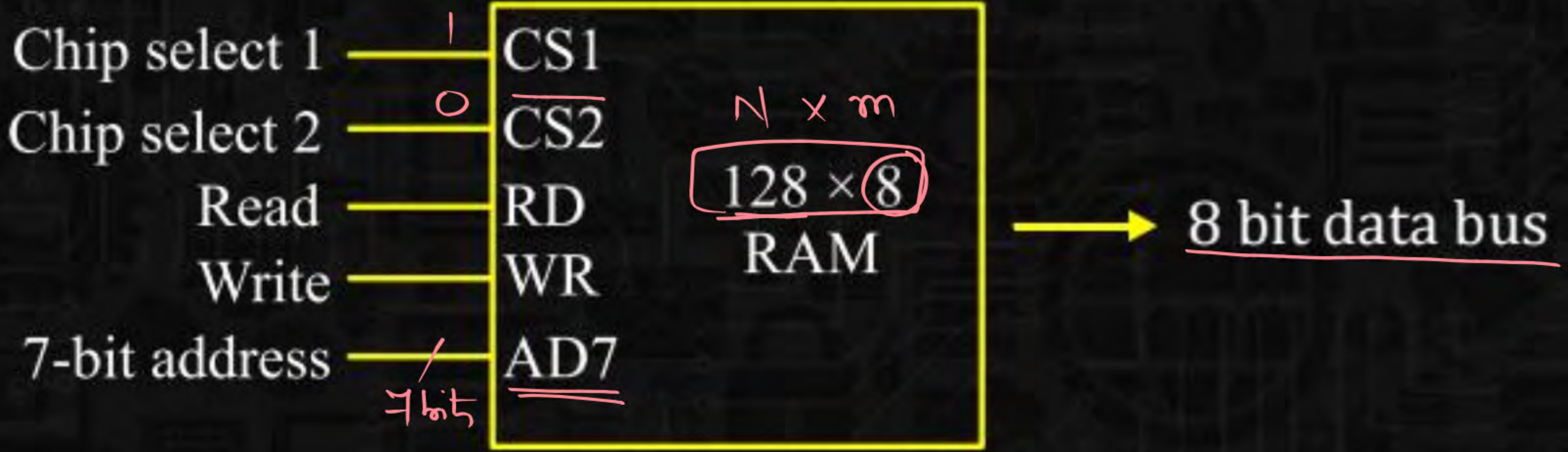
(j) $N = 64K$ bit

$m = (32 \text{ bit})\ 1W = 4B$

$N_W = \frac{64K \text{ bit}}{32 \text{ bit}} = 2KW$

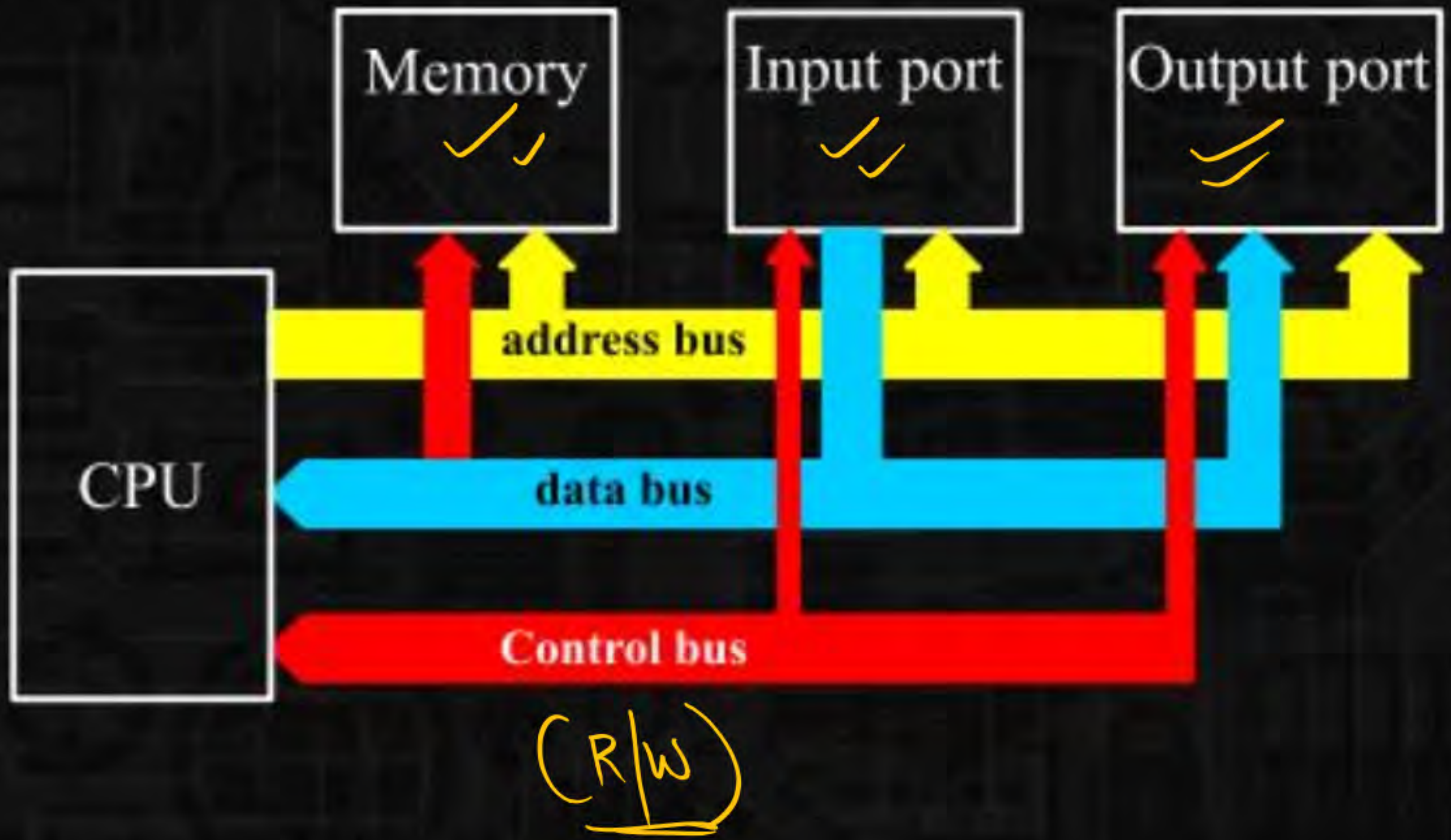
$N_{words} = 2KW$

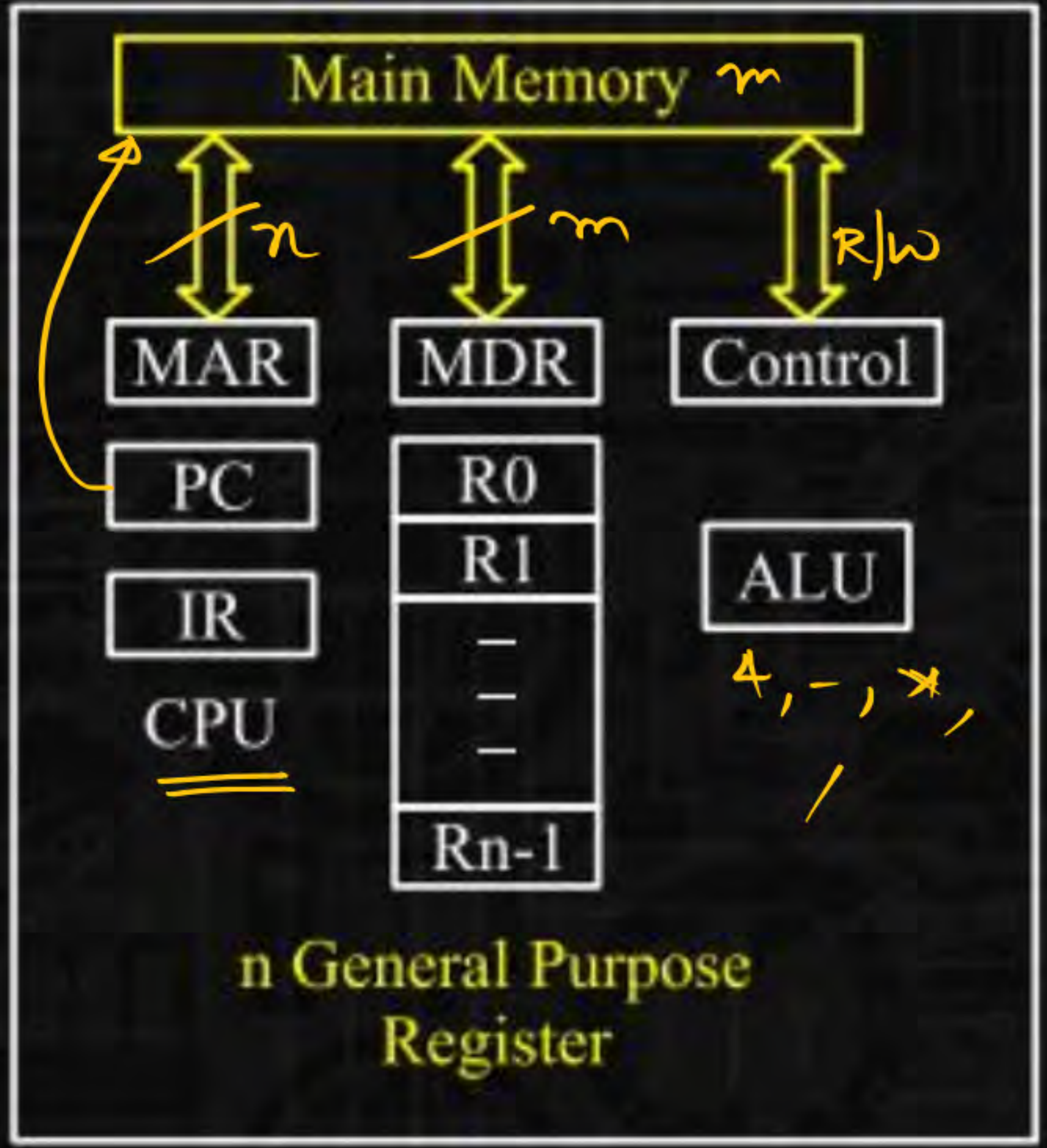
$N_{Bytes} = 2K * 4B = 8KB$



Block diagram of typical RAM chip

Memory Interfacing to CPU





Loading vs Linking:

I. **Loading** : activity of bringing the Program from Disk (Sec. Storage) to Main Memory (RAM) ; [Loader]

Static <Before R.T>

M.M: 70KB

Space Ineff.

Time efficient

Dynamic <During R.T>

→ Load only reqd Modules @ R.T

10KB

Space efficiency

Time inefficient

```
main( )          f( )        g( )        h( )
{                {           {           {
  :                :           :           :
  if (Cond)        g()         h();        scanf();
    f();           :           :         }
  :              }           }
}
```

main: 10KB, f: 15KB, g: 5KB, h: 12KB, scanf: 28KB

Total Prog. Size (on disk) = 70KB

II. LINKING : Carries out the activity of resolving external References;

- function calls
- Global variables
- objects, ...

$cc t1.c

KK.c

```
#include< >
....
main()
10KB {
   :
   BSA — ;
   f();
   :
}

f() : 5KB
{
   :
   BSA — g();
}

g() : 5KB
{
   :
   BSA — h();
   :
}
```

t1.c

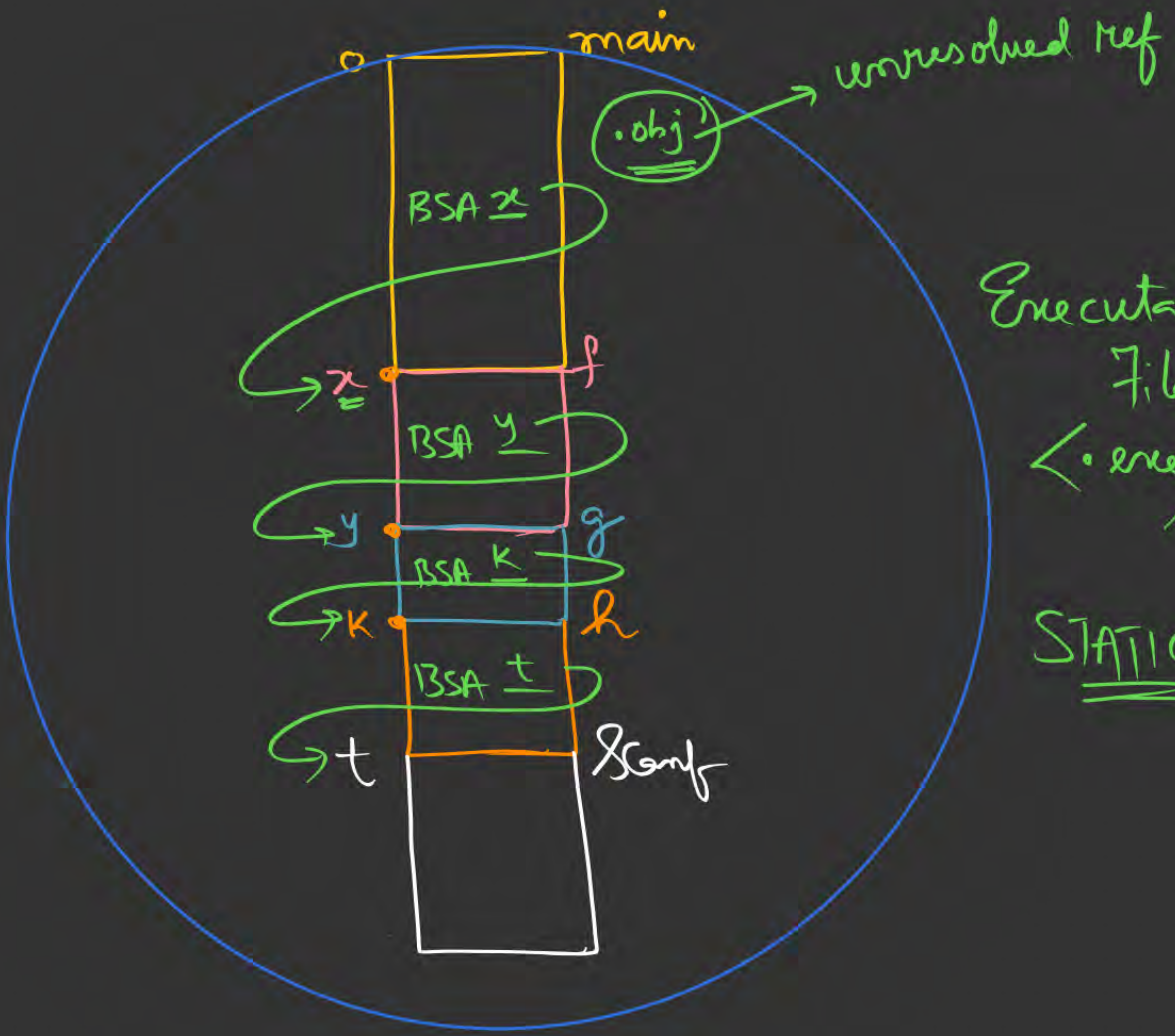
```
h() : 20KB
{
   :
   BSA — scanf();   10KB
}
```

Prog-Size : 50KB

(Multi-file-Program)

→ Linking

- Static (Before R.T)
- Dynamic (During R.T)



Executable File
<.exe>

STATIC LINKED

→ Space Inefficiency

→ Time efficient

BSA: Branch & Save Address


THANK YOU GW
SOLDIERS !