CS & IT ENGINEERING

Operating Systems

Memory Management

Lecture No. 2

By- Dr. Khaleel Khan Sir
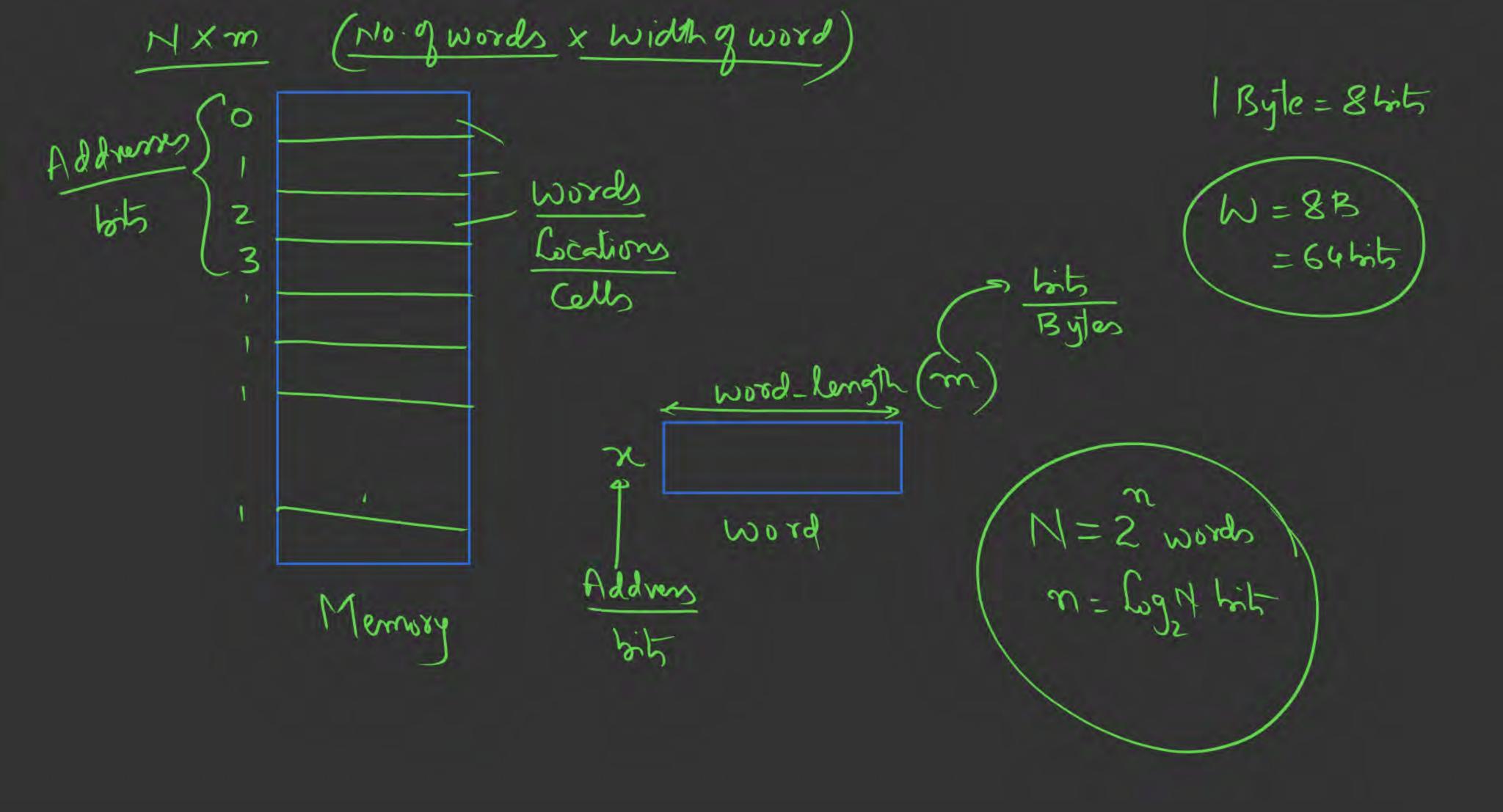
$N \times m$ (No. of words $\times$ width of word)

$$1 \text{ Byte} = 8 \text{ bits}$$

Addresses bits

$\left\{ \begin{array}{l} 0 \\ 1 \\ 2 \\ 3 \\ , \\ , \\ , \\ , \end{array} \right.$

Words
Locations
Cells

$$W = 8B$$
$$= 64 \text{ bits}$$

bits / Bytes

word-length (m)

x

word

Memory

Address bits

$$N = 2^n \text{ words}$$
$$n = \log_2 N \text{ bits}$$

Addressability & Capacity of Memory $= \dfrac{words \checkmark}{Bytes \checkmark}$ over $Bits$
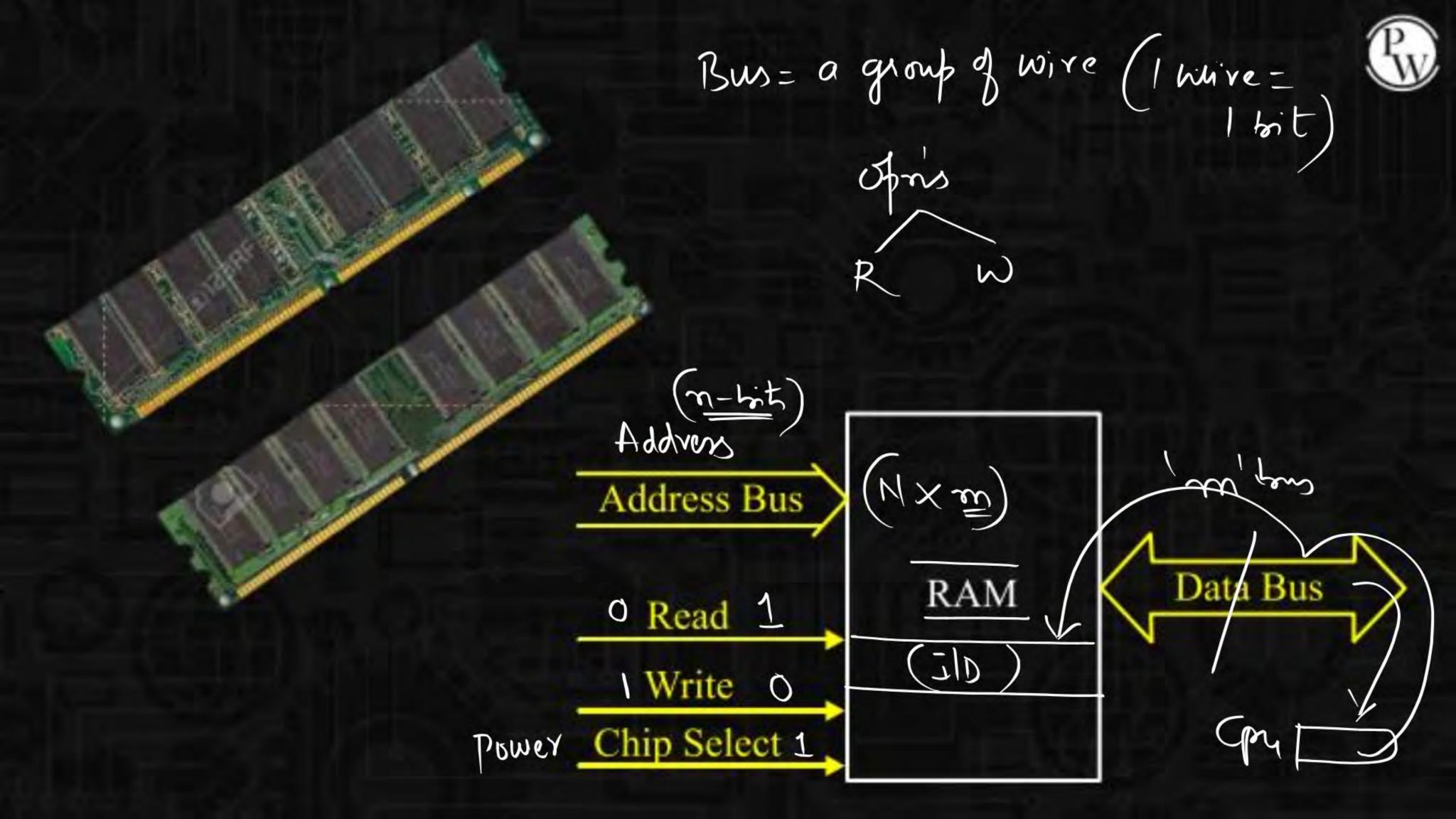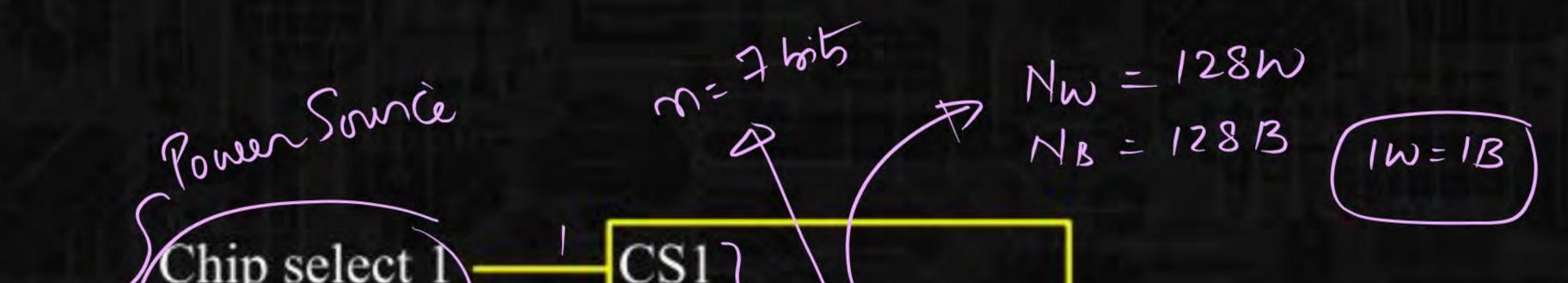
By default $1w = 1 Byte$

Practical
Unit of
Memory $= Bytes$

1) $n = 18 bits$

$m = (24 bits) = 3B$

$-24 bits- \quad 2^{18} W \checkmark$

$256 KW$

$\downarrow\!\!\downarrow$

$256 K * 3B$

2) $N = \overline{256 KB}$

$m = 4B (32 bits)$

$N_{words} = \dfrac{256 KB}{4 B}$

$= 64 KW$

$N = 16 B$

$m = 4B$

$N_W = \dfrac{16B}{4B} =$

$= 4W$

$$\overline{512m \times 16B} = 2^{29} \times 2^{4} = 2^{33} = \underline{\underline{8GB}}$$

3) $N = \underline{64G \text{ bits}}\checkmark$

$m = \underline{128 \text{ bits}}; \boxed{1W = 16B}$

$$N_W = \frac{64G \text{ bits}}{128 \text{ bits}} = \frac{2^{36}}{2^{7}} = 2^{29} \, W = \underline{512mW}\checkmark$$

$$N_B = \frac{64G \text{ bits}}{8 \text{ bits}} = \underline{\underline{8GB}}\checkmark$$

$N = 8 \text{ bits}$

$m = 2 \text{ bits}$

$$N_W = \frac{8}{2} = \underline{4}$$

Bus = a group of wire (1 wire = 1 bit)

op'ns

R   W

(n-bit)
Address

Address Bus

$(N \times m)$

___
RAM

(-|D)

Data Bus

0 Read 1

1 Write 0

Power   Chip Select 1

CPU
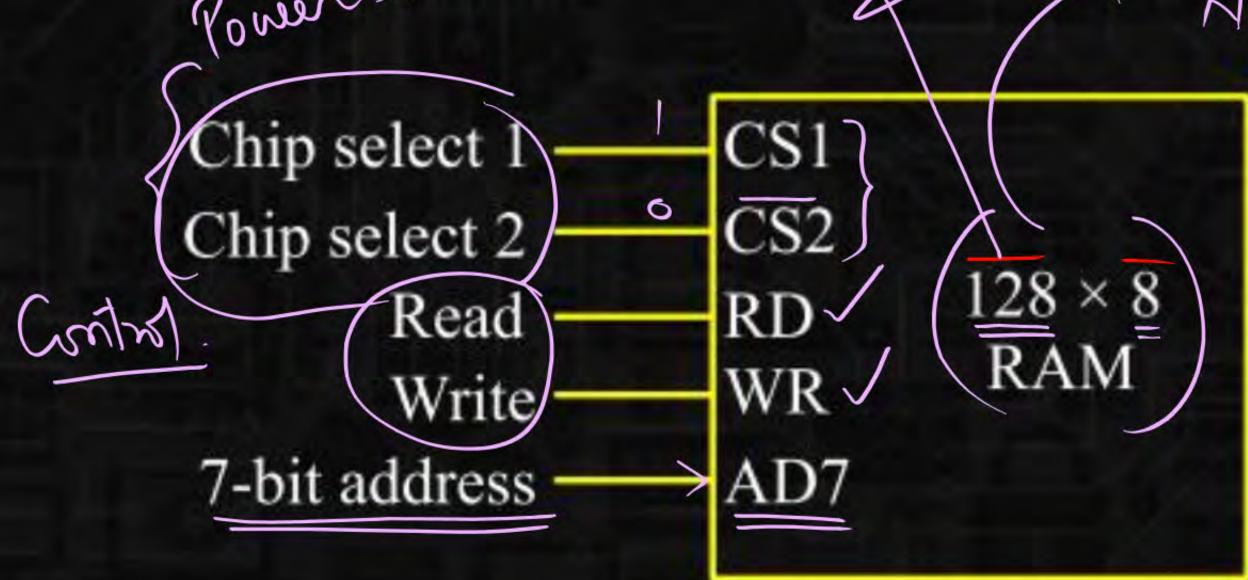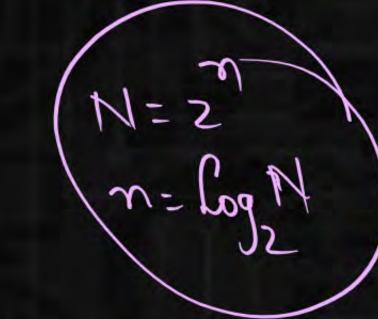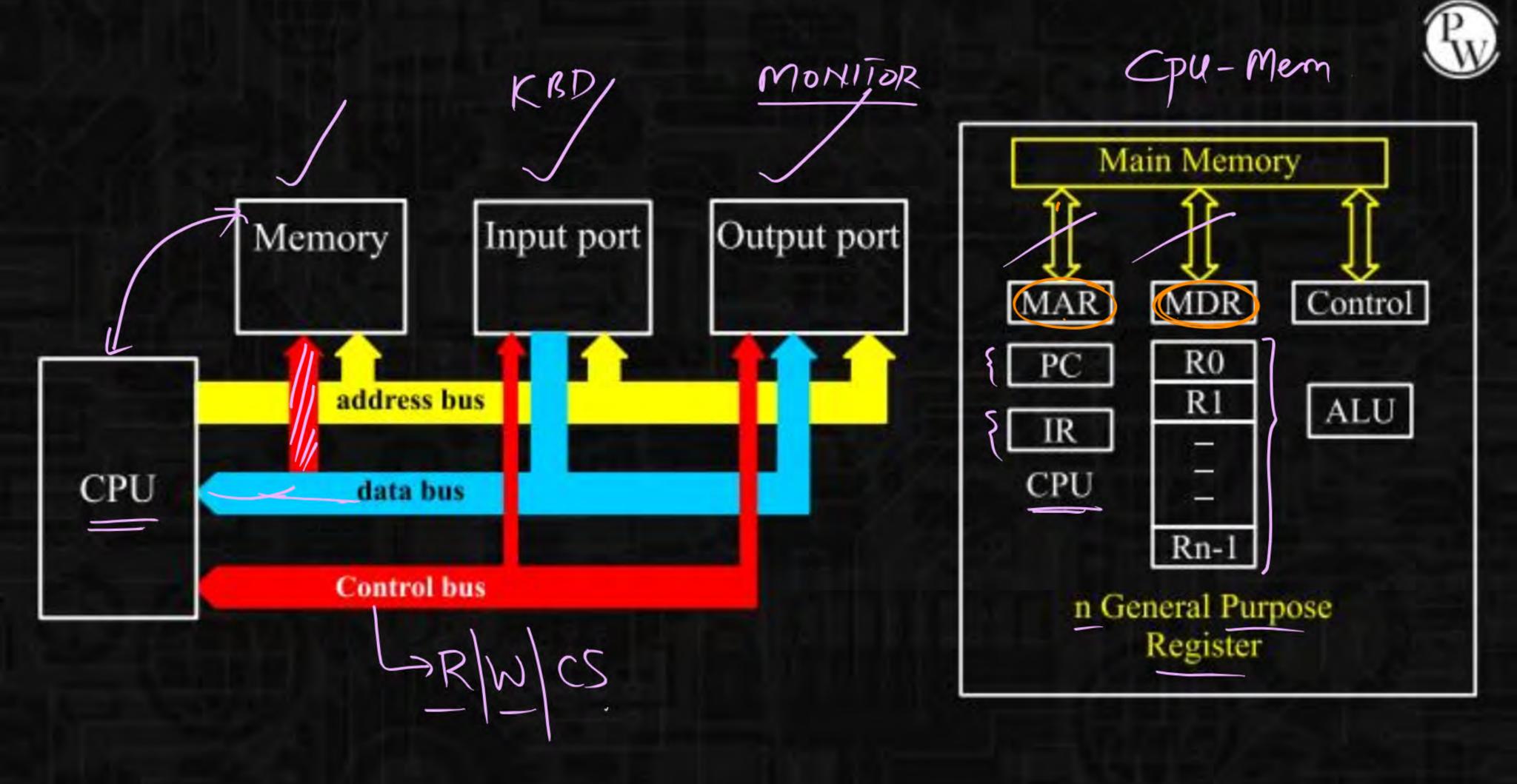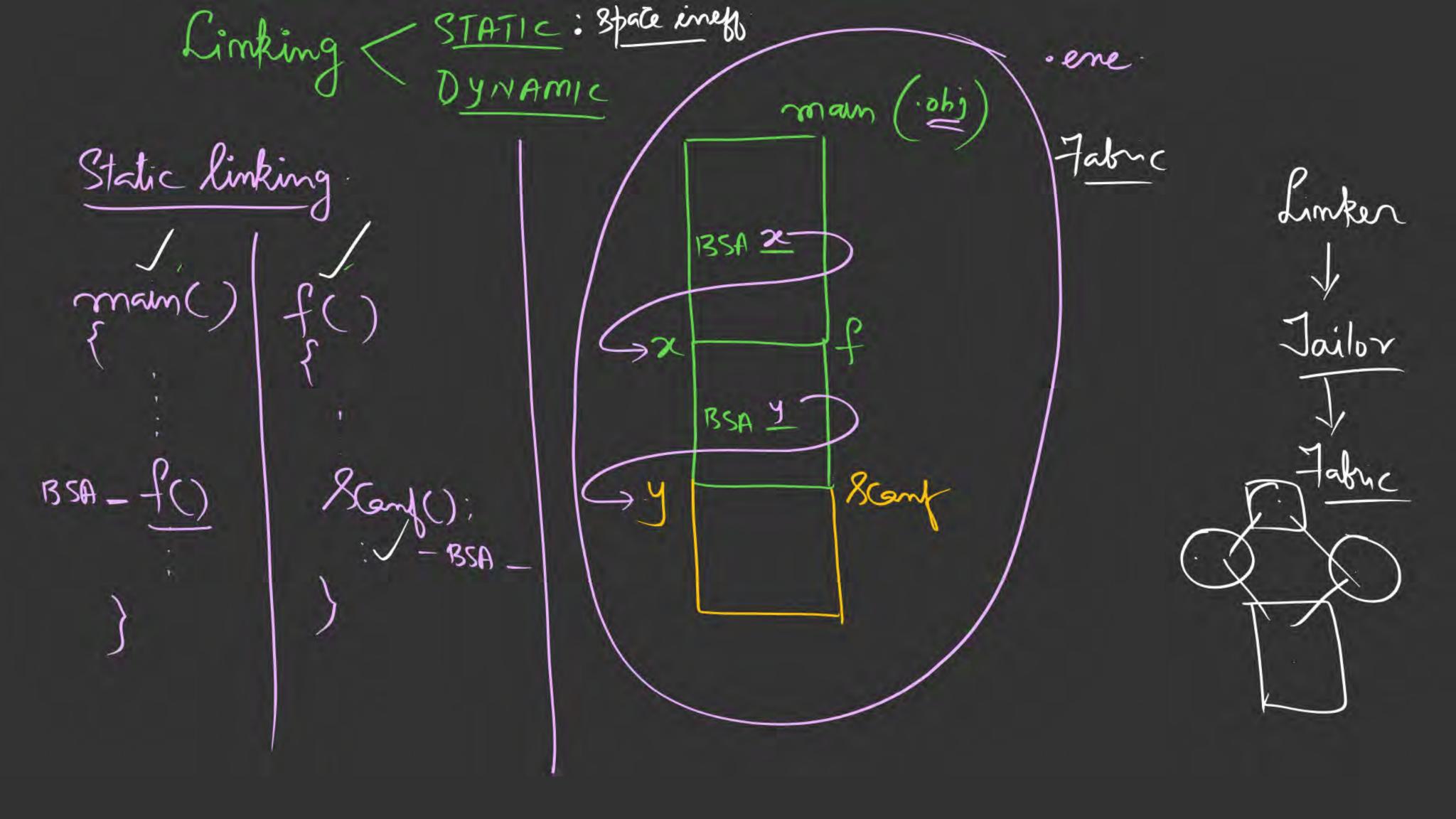
Block diagram of typical RAM chip

Power Source

Chip select 1 — CS1 ⎱ $n = 7$ bits $N_W = 128W$
Chip select 2 — CS2 ⎰ $N_B = 128B$

Control

Read — RD ✓ $\dfrac{128 \times 8}{\text{RAM}}$ ⟷ 8 bit data bus

Write — WR ✓

7-bit address → AD7

$1W = 1B$

$N = 2^n$

$n = \log_2 N$

KBD

MONITOR

CPU-Mem

Memory

Input port

Output port

Main Memory

MAR   MDR   Control

PC   R0

IR   R1

CPU   —

—

Rn-1

n General Purpose Register

CPU

address bus

data bus

Control bus

→ R|W|CS

Loading vs Linking

(Loader)

Loading :

Time ineff / Space efficient

Total Prog-Size (on disk) = 40 KB

Static   Dynamic → Loading on demand @ R.T

10KB
```
main()
{
    :
    iF
    y(Cond)
    f();
    :
    :
}
```

5KB
```
f()
{
    :
    :
    g();
    :
    :
}
```

15KB
```
g()
{
    :
    :
    h()
    :
    :
}
```

10KB
```
h()
{
    :
    :
}
```

Static Loading:

The whole Program is Loaded in Memory Before Starting the execution;

= 40KB of RAM

Adv : Faster execution (Time efficient)

Drawback : possible wastage of Space ⟨Ineffective utiliz of memory⟩

create
disk (.c) program
↓ Compiler
(.exe) Disk

┌─────┐
│ I │ D │
└─────┘
↓ Loading
(Memory)

Linking : is the process of resolving external references in the Program
(unresolved)

#include <stdio.h>

```
main()
{
    .
    :
    f();   BSA
    .
    :
    BSA = &(conf());
    .
    :
}
```

BSA : Branch & Save address

extern int x ;
```
f()
{
    .
    :
    .
    :
}
```

t.c

int x ;

my.c
```
f_1()
{
    .
    :
}

f_2()
{
    .
    :
}
```

t.c
```
main()
{
    f();
}
```

Compilation
↓
First line
↓
Execution
↓
Main()

Linking $\Bigg\langle$ STATIC : Space ineff
DYNAMIC

.exe.

main (.obj)

Fabric

Static linking

main()
{

    BSA — f()
    .
    .
}

f()
{
    .
    .
    Scanf();
    : — BSA —
}

BSA $\underline{x}$

x f

BSA $\underline{y}$

y Scanf

Linker
↓
Tailor
──
↓
Fabric

* Dynamic Linking : Linking @ R.T   [Late Binding]

10KB
.exe

Sp. efficient +
Time
ineff

```
main()
{
    .
    .
    .
BSA — f();
<STUB>;
    .
    .
}
```

```
f()
{
    .
    .
    Scanf();
    .
    BSA —
}
```

10KB

main

i

BSA 3 , <STUB>

R.T

STUB: a Small
piece of
Code

Linker
f()

.dll (dynamic
link
library)

(WIN)

3

Memory

Scanf
f()

D.L

Adv : 1) Space efficiency

2) Reusability

3) Modification (Flexibility) is easy

Drawback:

1) Time inefficiency

2) Less Security (Insecure)

Security

→ Static linking is more Security;

Library fns

(scanf, print)

Static Libr

(Dynamic Library)

.dll

```
   →
x ⟹ f()
   .      {

   .      }

   .  main()
   .      {
          .
          .
      BSA x; f();
          }   .
```

Address Binding : ⟨Association of Program Instn's & Data units to Mem locations (Addresss) is Address Binding⟩

Data unit

(int x ;)   declaration

x : variable : entity

Name   Type   Value   Addverss   Size
x int   (junk) (Mem.   (2B)
                              Loc.)

x [ ] 4000