⊛ Command to Create dynamic library :-

cc -shared -o libmno.so sum.o mul.o print.o

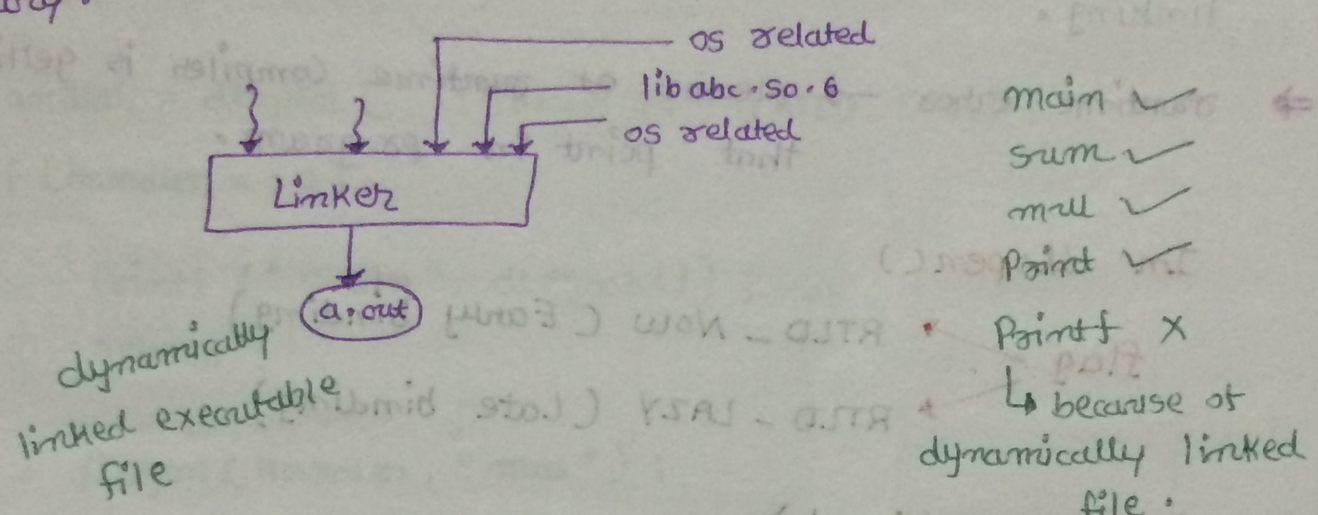
→ Dynamic library is seprately loaded into the file.

How to link with our own Created Static and dynamic libraries?

to link with static → cc -static main.c libabc.a

to link with dynamic → cc main.c -l libmno.so

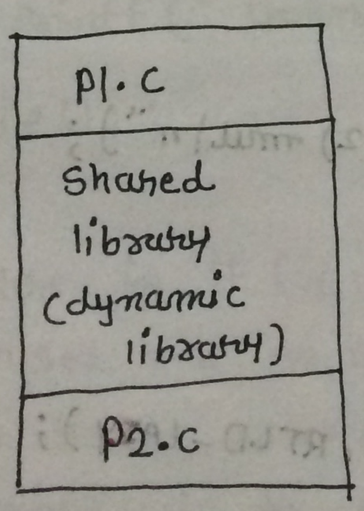
→ compiler will try to find for the def^n of any function inside the file and if not found them in predefined library.



⊛ advantage of dynamic library :-

→ You can save memory if multiple programs are using same thing by making it shared means we must put that thing into the dynamic library.

⊛ fixing the bugs are easy :-



→ Here P1.c and P2.c both are using this shared library.

→ Thus now we need to make changes in this library and then put it back means only updation no uninstallation.

→ But when uninstallation takes place data loss incase when we make opp^n using static library.

→ No data loss because no uninstallation.

* linking or binding is a concept related to calling function and called function.

* If calling function knows where exactly called function is present at compile time then the linkage | bondage b/w the calling function (main) and the called function (print) is called as static linking.

→ present at load time then linking between them is called dynamic load time linking.

→ present at runtime then is called dynamic runtime linking.

⇒ runtime error → means at runtime compiler is getting to that point in program.

In dlopen()

flag → RTLD_NOW (Early binding)
flag → RTLD_LAZY (Late binding)

```
#include <dlfcn.h>
#include <stdio.h>
main()
{
    int i=10, j=20, K, OP;
    void *handler;
    int (*p) (int, int)
    Printf (" Enter number---\n");
    Scanf (" %d %d ", &i, &j);
    Printf (" Enter the op\n 1) Sum\n 2) mul\n");
    Scanf (" %d", &op);
if (op==1)
{
    handler = dlopen (" ./libmma.so", RTLD_LAZY);
    if (handler==0)
    {
        printf (" %s\n", dlerror()) ;
        return ;
    }
```

```
    p = dlsym (handler, "sum");
    if (p==0)
    {
        printf ("%s\n", dlerror());
        return;
    }
    K = (*p)(i,j);   // function call using function pointer
    printf ("%d\n", K);
    dlclose (handler);
}
else if (op==2)
{
    handler = dlopen ("./libmno.so", RTLD_LAZY);
    if (handler==0)
    {
        printf ("%s\n", dlerror());
        return;
    }
    p = dlsym (handler, "mul");
    if (p==0)
    {
        printf ("%s\n", dlerror());
        return;
    }
    dlclose (handler);
}
else
    printf ("Unknown option ----\n");

while (1);
}
```

handler is holding the starting address of the dynamic library libmno.so.

for compile

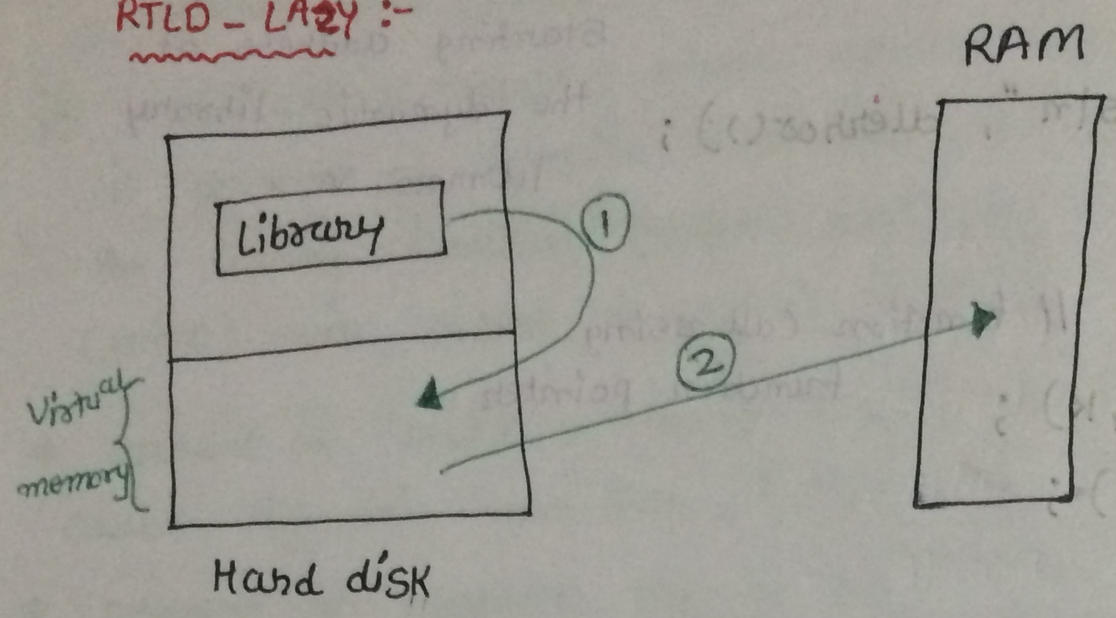
cc filename.c -ldl

linkwith -ldl

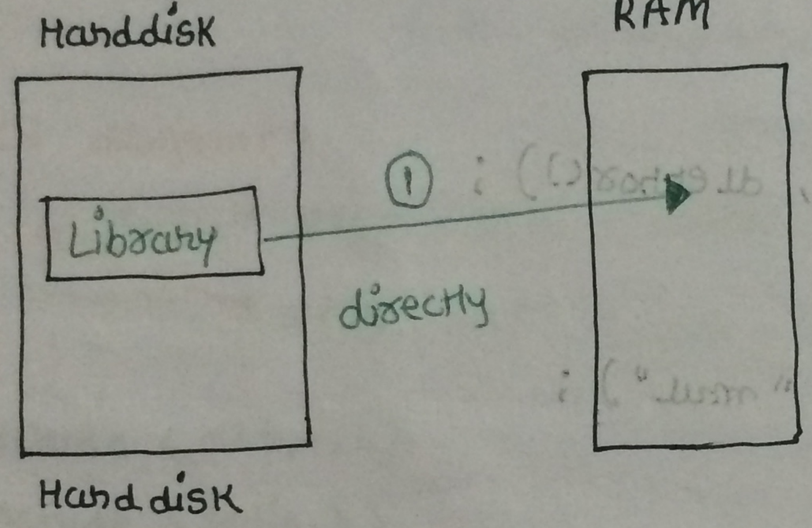
→ Here in if (op==1), dlopen, dlsym, dlerror and dlclose is used and so that inside if (op==2), thus here if user enter op==1 or op==2, then only the dynamic library will be loaded into the RAM and then at end it is closed and after the execution of the program at end it goes into while(1) loop, thus here we can say that now no need of library to present in the RAM unnecessarily, thus by dynamic runtime, linking we

Can archive it.

RTLD - LAZY :-



RTLD - Now :-



OS Concepts with Linux :-

1) process and process managment
2) File and file managment
3) memory and memory managment
4) Signal Handling
5) IPC (Inter process communication)
6) process Synchronization
7) Threads
8) Shall scripting

① what is O.S ?

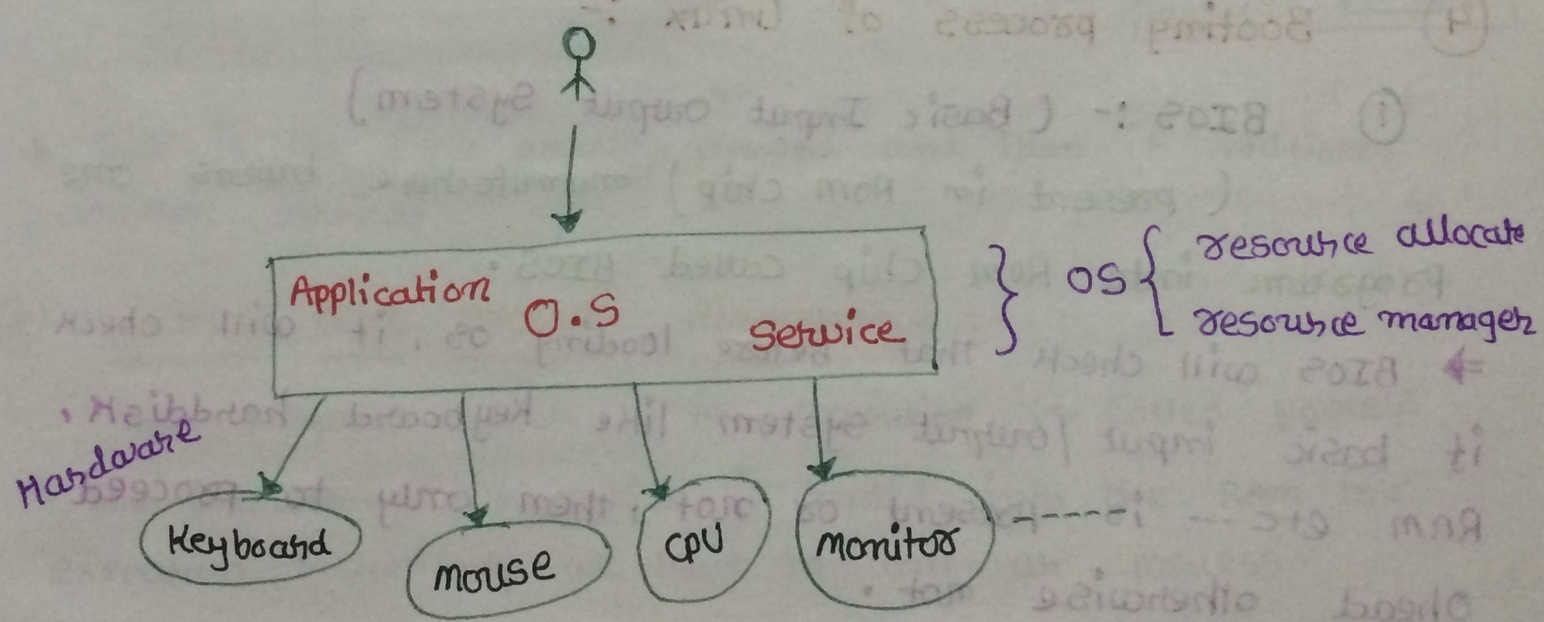
⇒ OS is a Software

Qualities :-

→ multitasking

→ manage the resources effectively
↳ (hardware)

→ resource allocate / resource manager



* why linux ?

It will support for around 20 architecture and linux is free source code

window
Intel ARM

Embedded OS / Embedded linux

→ OS is a software that control the hardware.

OS

Applications (optional)

Services (Kernel)
- process managment
- memory managment
- signal managment
- N/w managment
- Theard managment (compulory)

Service are sometimes drivers also.

② What are the components of OS?

1) Application
2) Services

} Drivers + Kernel

⇒ Kernel is called core of an operating system.

⇒ Application are optionals.

Services are mandetory.

④ Booting process of Linux :-

① BIOS :- (Basic Input output System)

(present in Rom chip) manufacture burns one program into Rom chip called BIOS.

⇒ BIOS will check that before loading OS, it will check if basic input/output system like Keyboard, harddisk, RAM etc --- is present or not, then only to proceed ahead otherwise not.

② if everything okay :-

Then bios will run one program called MBR (master Boot record) (a file of 12 bytes) present in the starting portion of HDD. (hard disk).

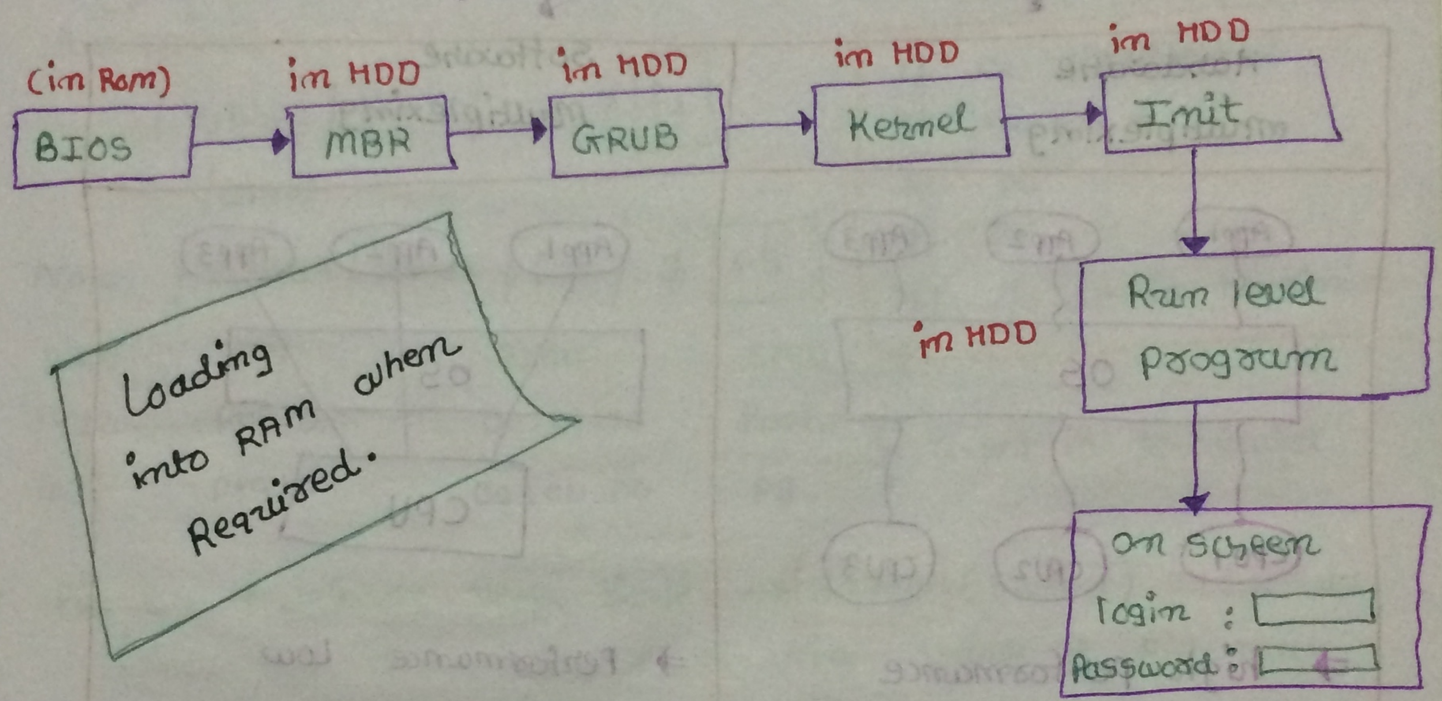
③ WBR executes GRUB (GRand Unified Boot Loader).

④ GRUB will load kernel.

⑤ Kernel loads all the services.
(Load some tempory file system mount the file system.)

⑥ Its starts one service called init.

⑦ Run level program :-
init process initiates all services.
All the services (drivers) are loaded into the RAM.

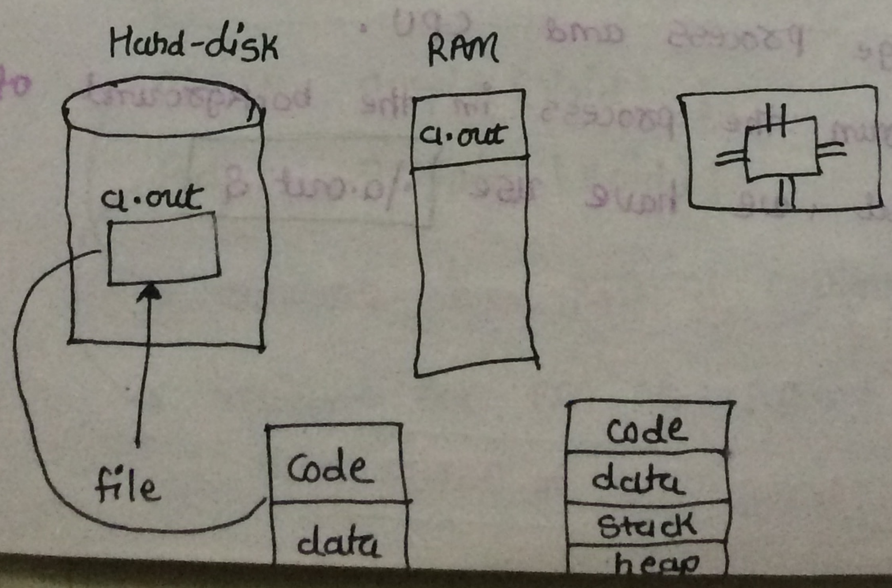


→ init is the first process created and then it initiates another services.

⊛ Process and process Managment :-

The program which is in execution is called process

→ The executable file which is loaded into the RAM for execution technically, we have to call as process.



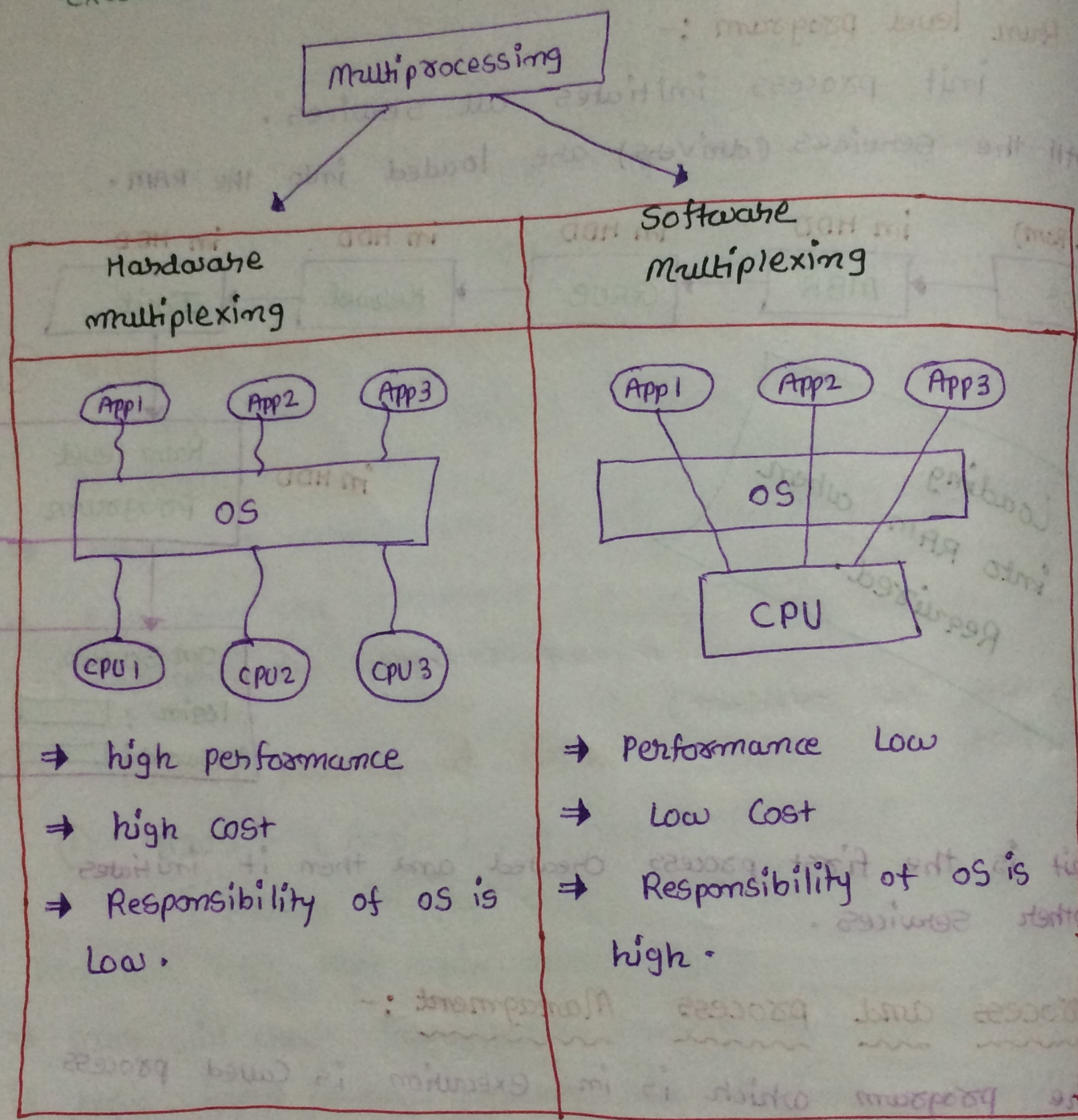
Then CPU fetches instruction and it will execute the program.

→ Life a process ends when the whole program ends.

⇒ Life of an executable file ends when we intentionally remove it by "rm" command or overwrite it.

⇒ cpu execute the instruction as all the code present in the RAM.

⇒ If multiple processing environment, multiple process executes simultaneously.



⊛ Process Manager :-

⇒ Job is to manage process and CPU.

⇒ If we want to run the process in the background of the current terminal, we have use **./a.out &**