It is rumoured that when they shut down the IBM 7094 at MIT in 1973 they found a low priority process that had been submitted in 1967 and not yet been run.

One solution to this problem is aging.

Gradually increase the priority of the processes that wait in the system for a long time.

Round robin algorithm

This algorithm is designed specifically for time sharing system.

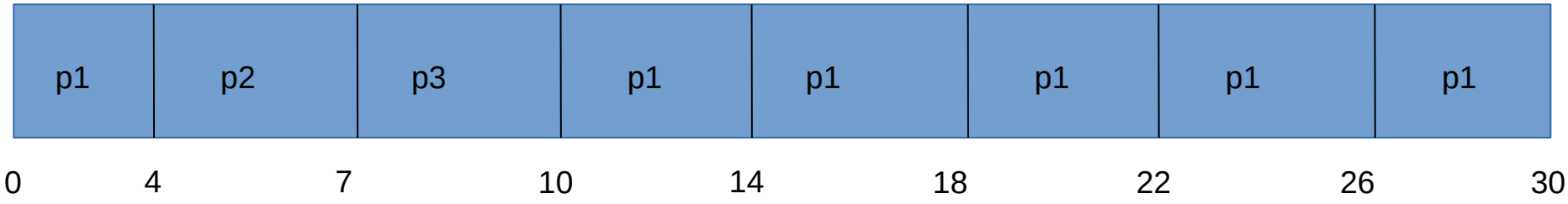
A small unit of time called a time slice or quantum is defined.

The ready queue is treated as a circular queue and the CPU goes around this queue, allocating the CPU to each process for a time interval of up to 1 time quantum .

New processes are added to the tail of the queue

Consider the following situation with arrival time 0

Process	CPU time	
P1	24	time slice =4
P2	3	
P3	3	



5.66

Time slice high and low???

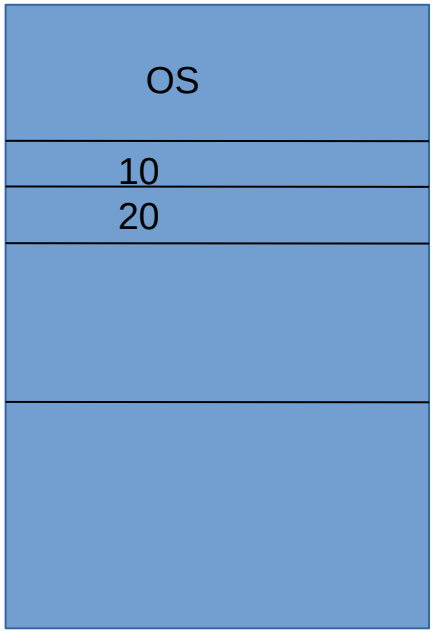
Memory Management

```
Int main()
{
    Int A,B;
    A=10;
    B=20;
    B=A+B;
    }
```

A	0X1234
B	0x1238

A and B are logical addresses and
0x1234 and 0x1238 are absolute addresses

Memory management unit of OS maps A----> 0x1234 and
B---->0x1238



Both the CPU and I/o system interact with the memory.

Before a user program is finally loaded into the memory and executed it goes through several stages.

Generally addresses in the source program are symbolic (such as A).

A compiler typically binds these symbolic addresses to relocatable addresses .

The linkage editor/loader will bind these relocatable addresses to the absolute addresses (such as 0x1234).

Each binding is nothing but a mapping from one address space to another.

Whatever may be the number of address bindings that may be required, the user program must finally be mapped to absolute address and loaded into memory to be executed.

A typical instruction execution cycle will fetch an instruction from memory.

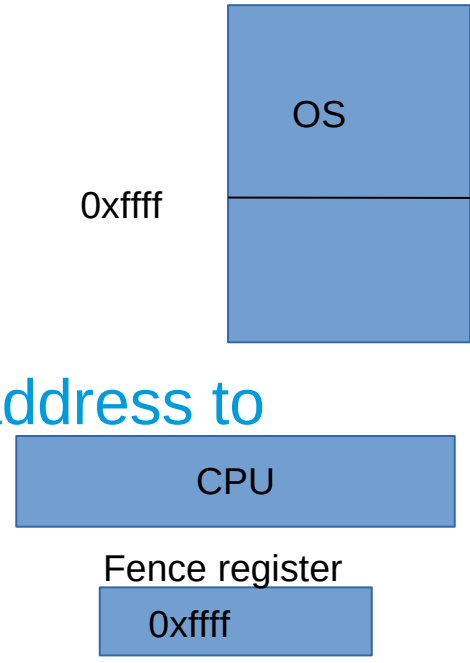
The instruction will be decoded and it may result in fetching the operands from the memory.

After executing the instruction it may also require storing of the result in the memory.

To protect the user from accessing the system memory (part of the memory where the os is stored) a fence register is used.

The fence register stores the fence address.

Every memory reference is checked against this this address to verify that it is indeed a legal memory reference.

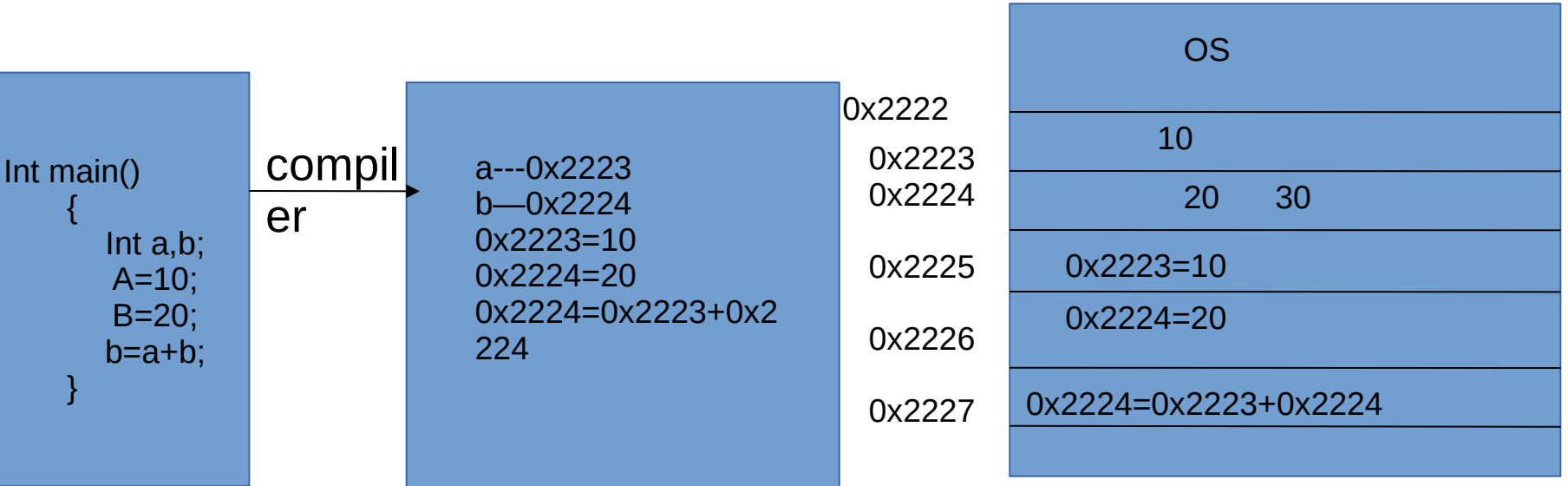


There may be 3 ways of mapping the user program to absolute address

- **Binding at compile time**

If the fence address is known at compile time , absolute code can be generated then

It only requires for the code to be loaded and executed.



Any disadvantage

If the fence address changes it is required to recompile.

Binding at load time

Here the compiler produces only relocatable code

Binding to absolute address is done during load time .

In case of changing in fence address the program only needs to be reloaded.