

Memory protection

Can be provided by software.

What is an example?

A system that keeps programmers completely away from direct access to memory addresses.

Alternatively, a check of every address by an instruction filter.

But far more efficiently and safely done by hardware.

Two requirements

Operating modes and privileged instructions

Limited address range

Provide hardware support to differentiate between at least two modes of operation.

1. *User mode* – execution done on behalf of a user.
2. *Kernel mode* (also *monitor mode*, *supervisor mode*, *privileged mode* or *system mode*) – execution done on behalf of operating system.

Why do we need both?

If we had modes and privileged instructions
but full memory access

Obviously no memory protection
but also no protection of the privileged instructions
- put any code you want in the system areas of memory

If we could limit memory accesses but there
were no modes and privileged instructions

Instructions are used to set up the memory management
registers.

If these are not privileged a user can change the area of
memory available.

Processor modes

Mode bit

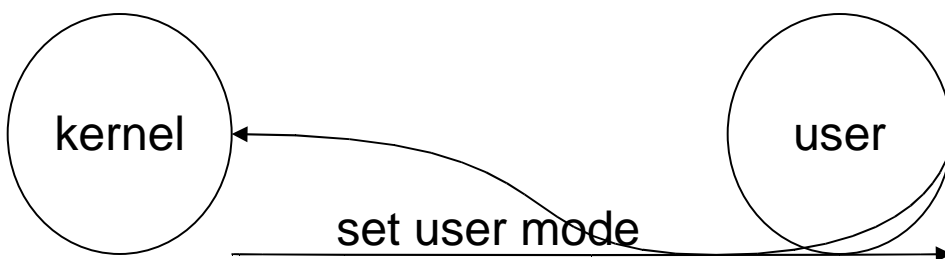
Added to the hardware processor status register to indicate which mode the processor is operating in.

Interrupts, faults, system calls cause the processor to change mode and ...

jump to a particular location.

Privileged instructions cannot be executed in user mode.

Interrupt/fault/system call



Memory protection

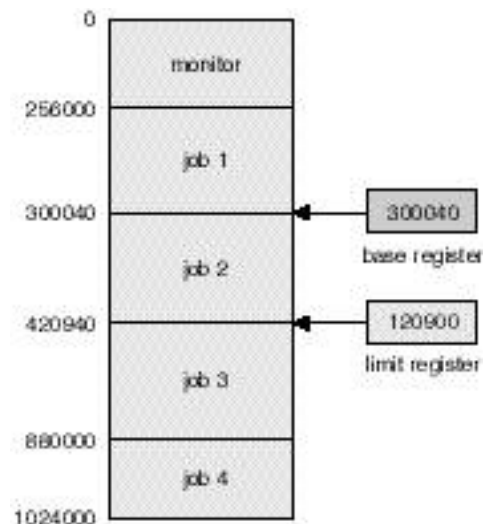
Each process gets allocated an area of memory which it can access.

All accesses outside that memory cause an exception (or fault).

- fixed size partitions
 - processes were designed to load in a particular partition
- base-limit registers
- memory pages

Devices can be protected using

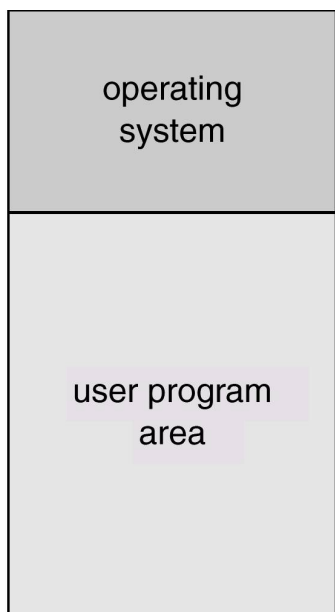
memory protection
or privileged instructions



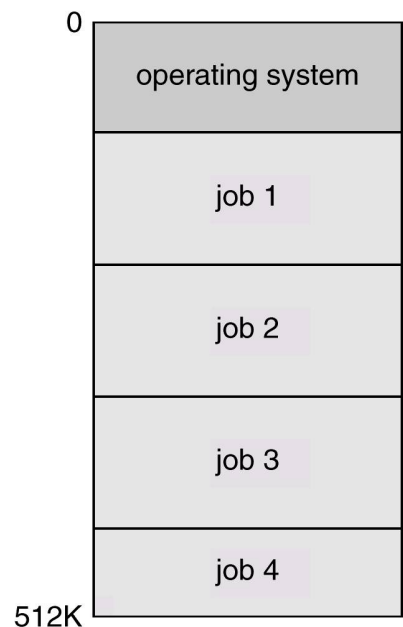
Batch systems

With memory protection and processor modes we can safely have multiple programs in memory.

from



to



Batch system innovations

Each job had its own protected memory.

Disks with file systems.

Files were associated with owners.

Scheduling was automated.

The aim was to effectively utilise all of the expensive hardware.

Computer operators now too slow.

Individual jobs could be suspended or killed.

Computer operators had consoles.

Maybe even VDUs.

Accounting could now be done automatically.

But from the programmer's point of view
nothing much had changed.

Scheduling

To maximize the use of the computing machinery the OS needed to know:

- what and how many devices a process would use
- what and how many files a process would use
- how much output it was likely to produce
- how long it was expected to take

Some of this information was submitted on the control cards, e.g. the files to be used.

Some of this information was inferred by the job queue the job was submitted on.

Different queues meant different expected resource requirements. Jobs on different queues were scheduled differently.

Scheduling (cont.)

Queue	1	2	3	4
Processor time (in seconds) default	120	20	20	120
maximum	1200	120	20	2000
I/O time (in seconds) default	120	20	20	120
maximum	1200	120	20	2000
Printer (in lines) default	500	500	300	500
maximum	2400	1200	500	3600

This system was used at the university.

The user could specify limits up to the maximum allowed for a queue. If no limit was specified the default for that queue was applied.

- Queue 3 was the default. It was designed for jobs which were quite small and which would move through the system smoothly. They were not allowed to use magnetic or paper tape (hence they did not require operator intervention).
- Queue 2 jobs were allowed to use two tapes (input or output).
- Most large jobs went through Queue 1.
- Queue 4 was for very large jobs and these jobs were scheduled manually.

To stop users abusing the system, they were limited to two jobs in any one queue at a time.

Power to the people

Circa 1970s.

Hardware was cheaper. This meant two things:

- Computer systems could afford to be nicer to users.
- People were now the expensive bit, we wanted to make them more productive.

Give them all a console

- Early versions used teletypewriters.
- But CRT VDUs weren't far off.
- The early ones still looked like printers.
- Text was displayed on the screen top to bottom / left to right.
- You couldn't easily edit in place.
- Flushing the whole screen at 2400 baud was a bit obvious.
- Side effect - people do different things with computers
- Shock, horror! Word processing.

Time-sharing systems

People want good response

- waiting over 1/5 a second is noticeable
- waiting over 5 seconds is unacceptable

People working at the machine are unpredictable

- they want to use files and devices at any time
- it is very hard for the OS to work out what processes will be compute intensive or IO intensive
- to service the same number of people more processes were active in the system

there was usually at least one process to deal with commands for each user, plus the normal work they requested

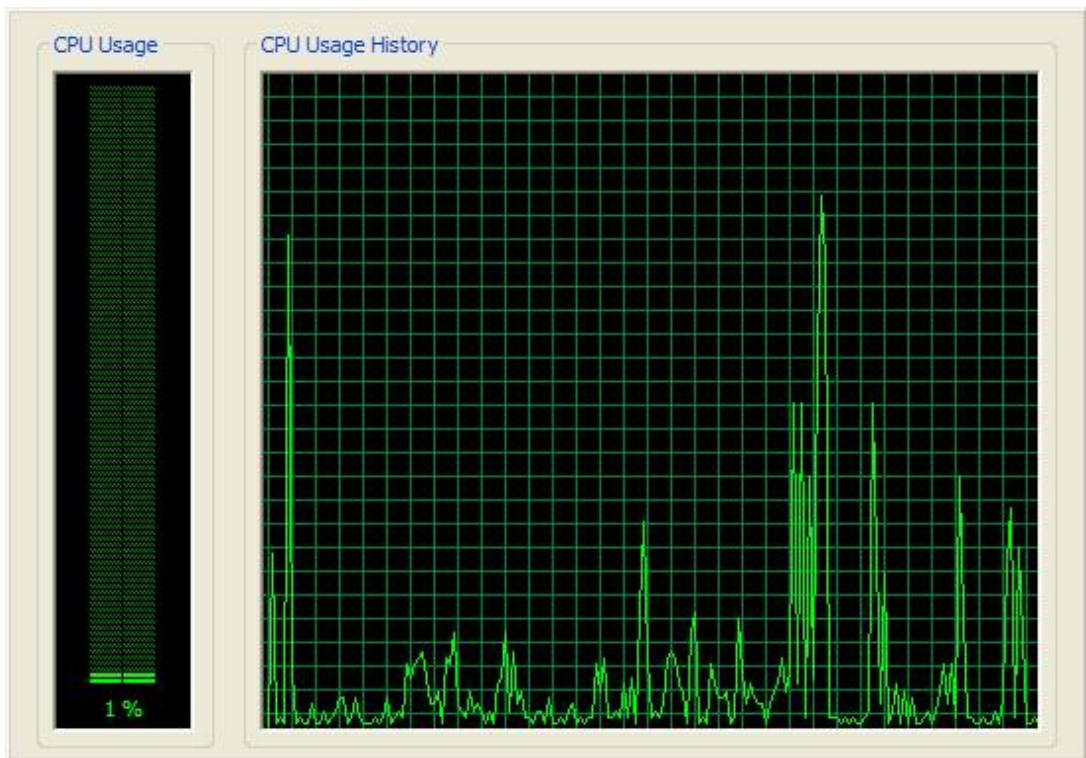
- the terminal was used for the entering of data and programs as well as JCL type commands
- because people were slow peripherals many more had to be online than jobs needed to be running on a batch system to get through the same amount of work
- users would also communicate with each other - early email systems and talk programs

Time-sharing systems (cont.)

Devices and the CPU couldn't be used to capacity otherwise the response time was too long.

So there had to be slack in the system.

- look at the processor usage on your Windows PC using System monitor or Task Manager
- or run `top` on UNIX



Time-sharing changes

Over-demand for resources can happen easily
(but we hope occasionally).

Most scheduling decisions are made by the
users.

Security becomes more of a problem

Hacking on a batch system was difficult.

The system must have some authentication process.

More user administration.

Profiles, defaults, resource limits

System administrators with more rights than “ordinary”
users.

Batch system remnants

Ways of running processes at specific times without user intervention e.g. `crontab`, `at` and Task Scheduler.

Until the arrival of cheap graphics terminals - the terminal still looked very like a card reader. e.g the `vt52`.

Command files or shell scripts - very like the control cards of a job control language.

Before the next lecture

Read textbook sections

- rest of chapter 1

- 3.1 System Components

- Read the assignment 1 handout. (And ask a question about it on the Wiki.)