Introduction to Computing

Knowing Your Computer

Malay Bhattacharyya

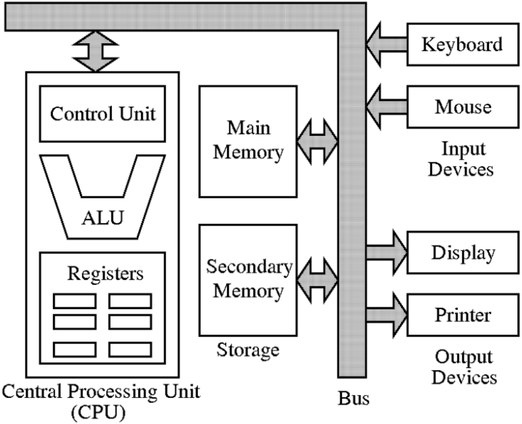
Associate Professor

MIU, CAIML, TIH
Indian Statistical Institute, Kolkata

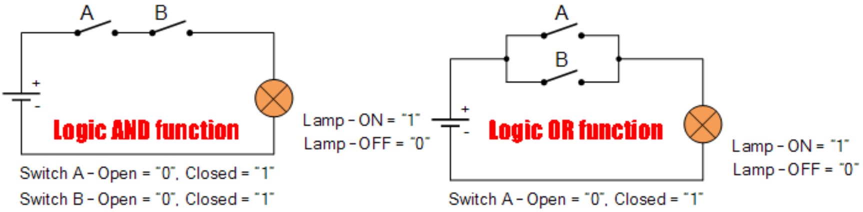
August, 2024

1. A Bit of Computer Architecture
2. Understanding the Hardware Management
3. Association between CPU and Memory

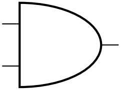
A digital computer – The internals



A digital computer – The circuitry

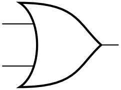


The Logic Gates – AND



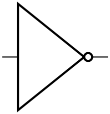
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

The Logic Gates – OR



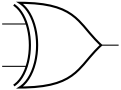
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

The Logic Gates – NOT



A	NOT A
0	1
1	0

The Logic Gates – XOR



$$\text{A XOR B} = (\text{A AND (NOT B)) OR ((NOT A) AND B)}$$

A	**B**	**A XOR B**
0	0	0
0	1	1
1	0	1
1	1	0

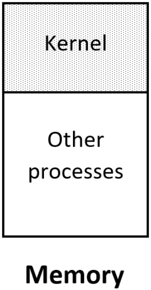
What is an Operating System?

Definition (Operating System (OS))

An OS is a software (a library of functions + set of programs) that manages a computer's hardware resources for its users and their applications.

Components of an OS

- Kernel: Core library that provides functions for basic operations (e.g., process creation / destruction) + interface to hardware via API (Application Programming Interface)
- Processes / programs
  - system processes – daemons/servers (httpd, lpd, sendmail, etc.)
  - user processes – shell, editor, compiler, utilities



The hardware resources managed by OS

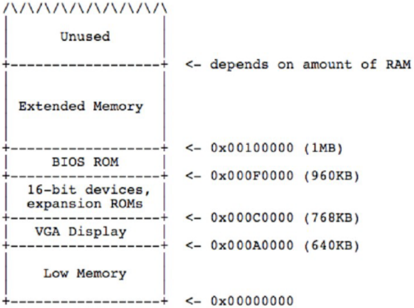
1. Processing units (CU, ALU, AGU, MMU, etc.)
2. Memory units
    - Temporary (RAM, etc.)
    - Permanent (ROM, HDD, etc.)
3. Input devices (Keyboard, Mouse, Scanner, etc.)
4. Output devices (Monitor, Printer, etc.)

Types of addresses

1. Memory addresses
2. I/O addresses
3. Memory mapped I/O addresses

Memory addresses

- Range: 0 to RAM size ($2^{32}-1$ or $2^{64}-1$)
- Mapping of main memory
  - Used to store data for code, heap, stack, OS, etc.
- Mapping of low and extended memory
  - Used by video buffers, expansion ROMS, BIOS ROMs, etc.
  - Used by latest OSs
- Accessed by load/store instructions



I/O addresses

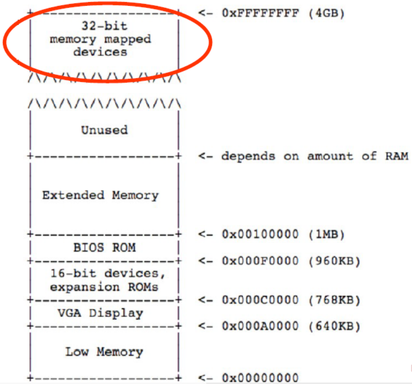
- Range: 0 to $2^{16} - 1$
- Used to access devices
- Uses a different bus compared to RAM memory access
  - Completely isolated from memory
- Accessed by in/out instructions

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

See: `https://bochs.sourceforge.io/techspec/PORTS.LST`

Memory mapped I/O addresses

- Provides additional space
- Devices and RAM share the same address space
- Instructions used to access RAM can also be used to access devices.
  - E.g., load/store



Who defines address ranges?

- Standard components
  - Industry standards (e.g., IBM PC standard)
  - Fixed for all PCs
  - Ensures BIOS and OS to be portable across platforms
- Plug and Play devices
  - Address range set by BIOS or OS
  - A device address range may vary every time the system is restarted

Why do we need memory?

- Non-volatile memory: Secondary
  – HDD, SSD, Flash Drive, etc.
- Volatile memory: Primary
  – RAM, Registers, Cache, etc.

Why do we need to manage memory?

The processes, together with the data they access, must be (at least partially) in main memory during execution.

For a better performance, we must share memory (i.e., keep many processes in memory). Memory management schemes control the way a shared memory is managed alongside the other tasks.

Selection of a memory-management scheme for a system depends on the system's hardware design.

The association between CPU and memory

The CPU fetches instructions from memory according to the value of the program counter.

Machine instructions can take memory addresses as arguments but not disk addresses. Therefore, any instructions in execution, and any data being used by the instructions, must be in one of these direct-access storage devices.

Note: If the data is not in memory, they must be moved there before the CPU can operate on them.

Memory address space

We need to make sure that each process has a separate memory space. Separate per-process memory space protects the processes from each other and is fundamental to having multiple processes loaded in memory for concurrent execution.

Memory consists of a large array of bytes, each with its own address. A base and a limit register define a logical address space.

Address binding

High-level code:

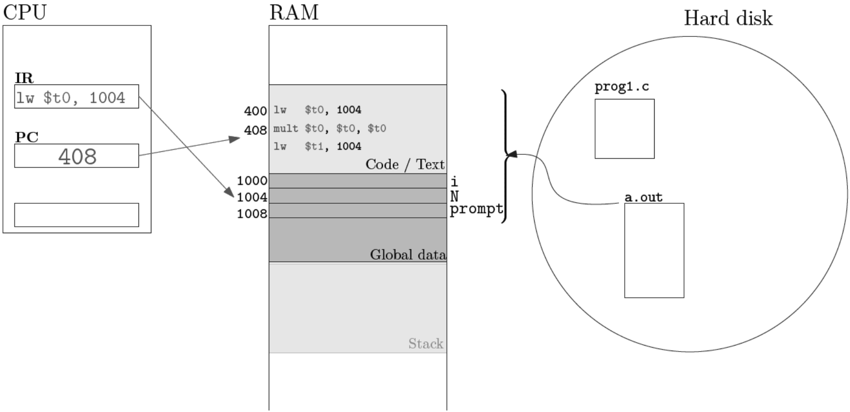
```
char prompt[] = "i";
int N = 20;
int i;
i = N*N + 3*N;
```

$\Longrightarrow$

MIPS machine instructions:

```
lw $t0, 1004 # fetch N
mult $t0, $t0, $t0 # N*N
lw $t1, 1004 # fetch N
ori $t2, $zero, 3 # 3
mult $t1, $t1, $t2 # 3*N
add $t2, $t0, $t1 # N*N + 3*N
sw $t2, 1000 # i = N*N + 3*N
```

Address binding



* This figure shows a hypothetical case, however, the actual organization of contents in the RAM may vary for different compilers.