INTERNAL HARDWARE

Processor

Device that carries out computation on data by following instructions in order to produce an output.

Chip: an electronic component contained within a thin slice of silicon.

Main memory

Memory is a medium of storage. The main memory is composed mainly by RAM and ROM:

- RAM: Random Access Memory, stores data and can be read from and written to
    - Volatile → content must be refreshed cyclically by memory controller and content lost if power is withdrawn
- ROM: Read Only Memory, stores data that normally can only be read from. It is non-volatile and it holds the BIOS (Basic Input Output System) used to set up the computer.

Buses

Microscopic parallel wires (etchings) that transmit data between internal components. Each parallel etching for a bus must be exactly the same length. A bus can transmit a single word at a time, its length is the same of the bus width (number of etchings).

- Data bus: transfer data between the processor, memory and I/O controllers.

  More etchings → more data can be transmitted per clock cycle → faster computer

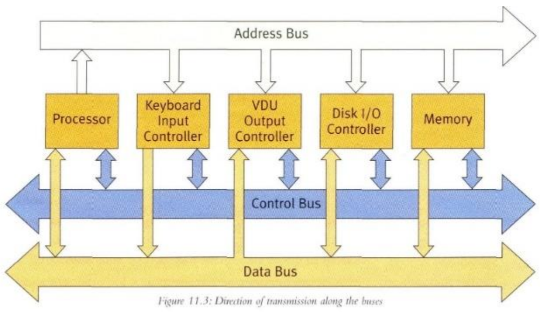


Figure 11.3: Direction of transmission along the buses

- Address bus: used to specify a physical address in memory so that the data bus can access it. One way: processor to memory.

  More etchings → more unique addresses → support more memory

- Control bus: controls the flow of data between the processor and other parts of the computer.
  Not much difference changing the wide of the control bus because there are not many controls

Peripherals

Any device not connected directly to the processor & memory. Peripherals connect via I/O Controllers.

I/O Controller: is an electronic circuit that:

- Controls the flow of information between the Processor & the Input & Output devices
- Translates signals from / to I/O devices to those understood by the Processor
- Effectively acts as a buffer to accommodate the varying speeds of different I/O devices.

Processor Architecture

Von Neumann: a technique for building a processor where data & instructions are stored in the same memory and accessed by buses. Just one address bus and data bus shared between instructions & data.

Harvard: a technique for building a processor that uses separate buses and memory for data & instructions.

- reduce bottleneck of single data/address bus, so reduce delays waiting for memory fetches.
- avoids possibility of data being executed as code (used by hackers)

Harvard architecture is used in specific use computer systems, while the Von Neumann model in general purpose PCs.

FETCH EXECUTE CYCLE

Stored Program Concept

Von Neumann Architecture: both instructions and data are stored in memory together. Each instruction or unit of data is:

- Fetch: processor fetches the program's next instruction from memory. The instruction will be stored at a memory address.
- Decode: processor works out what the binary code at that address means
- Execute: processor carries out the instruction.

Fetch Execute Cycle
The continuous process carried out by the processor when running programs

Processor components:

Control Unit: part of the processor that manages the execution of instructions

Arithmetic Logic Unit: part of the processor that processes and manipulates data

Clock: device that generates a signal used to synchronise the processor components

Register: small section of temporary storage that is part of the processor. Stores data or control instruction during the fetch-decode-execute cycle.

Status Register, SR: stores the status of current operation (e.g. Overflow)

Interrupt Register, IR: type of status register that stores signals received from other components (e.g. I/O Controller from a printer / keyboard)

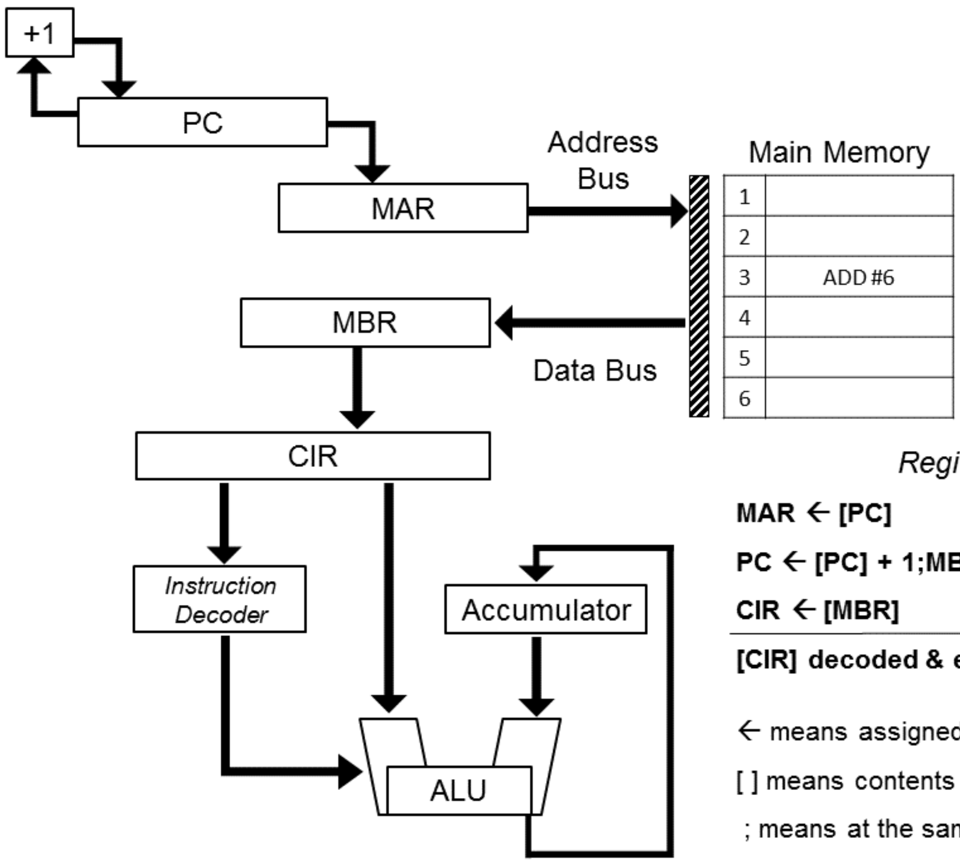
Current Instruction Register, CIR: stores instruction currently being executed by the processor

Program Counter, PC: stores the memory address (location) of the next instruction

Memory Data/Buffer Register; MDR/MBR: stores the data that has just been read from or about to be written to main memory

Memory Address Register, MAR: stores the memory address (location) where data in the MDR is about to be written in or read from

Fetch Execute Cycle



Register Notation

MAR ← [PC]

PC ← [PC] + 1;MBR ← [Memory]$_{\textbf{addressed}}$

CIR ← [MBR]

[CIR] decoded & executed

← means assigned to

[ ] means contents of

; means at the same time

The address contained in the PC is copied into the MAR that will allow it to be transferred via address bus to the main memory. Then while the PC is increased by one, the data or instruction at the addressed memory location is copied via data bus into the MBR. Then the content of the MBR is copied in the CIR.

Decoding an instruction

CIR holds the Instruction to be decoded, the instruction is broken into two parts

Op-Code: CIR refers to Instruction Set to see what this Op-Code is & informs the Control Unit, Control Unit sets switches to set up the ALU for this Op-Code

Operand: Data component of the instruction is passed to the ALU as an input, if applicable

Factors affecting processor performance:

- Clock speed: increasing the clock speed the computer executes instructions faster. Problems: heat and interference between subsequent signals.
- Bus width and word length: effect different for each bus.
- Multiple cores processor: runs faster but not at a multiple speed because different cores must be controlled by something.
  - Core: collection of processor components capable of running a fetch execute cycle
- Cache: temporary memory locations available to the processor on the chip. Not as fast as registers but much faster than Main Memory. Cache is used to store instructions and data that the processor predicts it will need soon to save cycles.

Interrupts

Interrupt: a signal sent by a device or program to the processor requesting its attention. Examples are user pressing a key, error in the execution of a program, power about to go off ecc.

The processor checks if an interrupt arrived at the end of each cycle by looking at the contents of the interrupt register.

If an interrupt has occurred the interrupt service routine (ISR) calls the routine to handle the interrupt. It does this by placing the contents of the registers on to the system stack, so that once the interrupt is handled the program can carry on.

Level	*Type*
1	Hardware failure
2	Reset interrupt
3	Program error
4	Timer
5	Input/Output

Vectored interrupt mechanism: each type of interrupt has an associated memory address (known as vector) that points to the starting address of the ISR.

Priorities: method for assigning importance to interrupts in order to process them in the right order. Processes with higher priority can interrupt the ones with lower priority, interrupts with same priority are dealt with on a first-come first-served basis.