1.1.1 Structure And Function Of The Processor

Revision sheet

1 Structure of the Processor

The processor is made up of a number of registers, buses and a number of control chips.

1.1 Arithmetic and Logic Unit

This performs all the arithmetic (addition, subtraction) operations as well as all the logical operations (branching).

1.2 Control Unit

This uses the clock pulses to synchronise the different components of the CPU. It decodes the program instruction (splitting the operator apart from the operand, then working out what the operator) in the CIR; then selects machine resources (eg, data source register and particular arithmetic operations). It then coordinates the activation of those resources.

1.3 Registers

Registers are a location in the processor, used for a particular purpose. They often temporarily store data or control information; they allow very high speed access.

1.3.1 Program Counter

This holds the address of the next instruction; unless there is a jump instruction, in which case the CIR sends the address of the jump instruction to the Program Counter. The Program Counter sends the address to the MAR.

1.3.2 Memory Address Register

This receives the address of the instruction or data from the Program Counter. It holds the address of the data or instruction to be used. It receives the operand (from the instruction) from the CIR.

1.3.3 Memory Data Register

This receives an instruction or data from a memory location in the MAR, or from a memory location in address part of the instruction / accumulator. It is a two-way register that holds data fetched from memory (which is ready for the CPU to process) or data waiting to be stored in memory.

1.3.4 Current Instruction Register

This contains the instruction to be executed. It splits the instruction into component parts and holds the opcode while it is decoded. It sends the address to the MAR for accessing data/value to send to the accumulator as well as sending the address (of the next instruction) to the PC in the event of a jump instruction. It also determines the type of addressing to be used.

1.3.5 Accumulator

This holds the output from any use of the ALU. It also holds and data inputted to the CPU as well as holding any outputs before they are removed from the CPU.

1.3.6 Index Register

Most commonly, an index register holds the current offset of a memory location with another register holding the bass address, so the combination of the two registers creates a completed memory address. One of the special functions of an index register is that it can be used to easily step through memory addresses either by being incremented or decremented as needed - so data structures such as arrays and stacks can be traversed.

1.3.7 Interrupt Register

This is checked when each cycle is completed by comparing the priority of the current task with the interrupt register.

1.4 Buses

A bus consists of a number of parallel wires (conductors); each wire handles a single data bit, so it is either logic 1 or logic 0.

1.4.1 Data Bus

Typically consists of 8/16/32/64 separate lines and provides a **bi-directional** path for moving data and instructions between system components. Width of it can affect CPU performance.

1.4.2 Address Bus

Used to carry the address of the required data and, to address the inputs and output ports during any input or output generation. Usually **unidirectional** as the CPU is the only thing that needs to generate addresses.

1.4.3 Control Bus

Transmits command timings and specific information between system components (for example: memory read & write, input/output read & write and clock signals to synchronise operations). It is **bi-directional**.

2 Fetch-Decode-Execute Cycle

2.1 Summary

Fetch The next instruction is fetched from main memory/address.
Decode The instruction is interpreted/translated/split into opcode and operand (in CIR).
Execute The appropriate instruction/opcode is carried out on the operand.

2.2 Detailed steps

- The program counter holds the address of the next instruction.
- Copy the contents of the program counter to the memory address register.
- Increment the program counter.
- Load instruction pointed to by the memory address register to the memory data register.
- Copy the instruction from the memory data register to the current instruction register.
- Decode the instruction in the current instruction register.
- Execute the instruction - fetch data from the address pointed to by the operand or process the data. If its a jump:
  - Change the contents of the program counter to address part of instruction.
  - Copy address part of instruction in current instruction register to program counter.

3 Factors affecting Processor performance

There are a number of factors which affect the performance of the CPU.

3.1 Bus Width

The width of the data bus can affect the speed. For example, if the data bus is 8 bits wide and the instruction is 16 bits wide, the processor must access the main memory twice just to fetch the instruction.

3.2 Cache

Cache is a very small amount of very fast memory inside or near the CPU. The most frequently used instructions are stored here so they can be accessed quicker. The size of the cache and proximity to the CPU is a key factor in CPU performance.

3.2.1 Types of Cache

There are a number of different types of cache, each with different speeds. The levels get progressively slower with increased numbers - L1, L2, L3 and L4.

3.3 Number Of Cores

Modern CPUs have multiple cores, each of which is able to process a different instruction simultaneously, with its own fetch-decode-execute cycle. The more cores, the more FDE cycles can take place at once, the faster the CPU.

3.4 Clock Speed

All of the processors activities begin on a clock pulse. The greater the clock speed, the paster the instructions will be executed. Clock speed is measured in Hertz (Hz) and is usually in GHz or less commonly MHz.

3.5 Pipelining

This is a technique used to improve processor performance. It is where in one clock cycle, one part of the CPU is doing fetch, another decoding and another executing. This means three processes can be handled concurrently in one core of the CPU. Pipelining avoids keeping internal components idle and means instructions are executed at a faster rate, which in turn improves processor performance.

3.6 Words And Word Size

This refers to the amount of data that can be handled at one time by the processor. Word length determines the size of a bit pattern that cen be transferred to or from main memory in one operation; the size of the processor registers (these are the same as the word length); the width of the data bus (is equal to the word length); size of each addressable memory location (typically, these are the same size as a word). The larger the word size, the greater the amount of data that can be transferred to the CPU in one pass.

4 Computer Architecture

There are two key designs of CPU - Harvard and Von Neumann. Both make use of the stored program concept - a program must be loaded into memory to be executed by the processor; instructions are fetched one at a time, decoded and executed sequentially by the processor, sequence can only be changed by conditional or unconditional branching.

4.1 Von Neumann Architecture

This uses the idea of storing program instructions and data in **shared** main memory and moving them between memory and the processor. It is used in many modern day computer systems. The processor can access the instructions and data in the main memory as required to execute the program. It does this by using buses (address bus to identify the addressed location and data bus to transfer the contents to and from the location). This means that the same address and data buses are used in the process of transferring instructions and data between main memory and the processor. A third bus, the control bus is used to synchronise and control operations.

4.2 Harvard Architecture

This keeps instructions and data in **separate** memories. The processor accesses these memories using separate data and address buses. Harvard architecture is commonly used in embedded systems, or microcontroller devices.

4.3 Comparison Of The Two

4.3.1 Concurrent Access

Harvard is characterised by the use of separate memory units and buses for instructions and data, meaning that both memories can be accessed simultaneously. Von Neumann uses the same address and data buses for both instructions and data, which means that both instructions and data share the same pathways.

4.3.2 Different Memory Types

Harvard - each memory can be adapted to fit the needs of a particular system; the instruction and data memories can be different sizes, word lengths, or use different types of technologies. Von Neumann only allows for instructions and data to be stored in the same size, word length and use the same technology; this can be exploited by hackers who can disguise instructions as data that the processor may unknowingly execute when attempting to read data.

4.3.3 Overall

Von Neumann architecture is used more commonly in situations where the variety of programs needed to be run are not known in advance. Harvard architecture is more commonly used in embedded systems or in systems where speed of operation is very important.