Starter

In your group, decide on your roles below. Then look at the instructions document for your role – do not look at the others!

- CPU
- Display
- Main memory
- AU (for groups of 4 only)

Extension:
Estimate your group's average clock speed

Processors 1

Specification

4.7.3.1 The processor and its components

Content

Explain the role and operation of a processor and its major components:

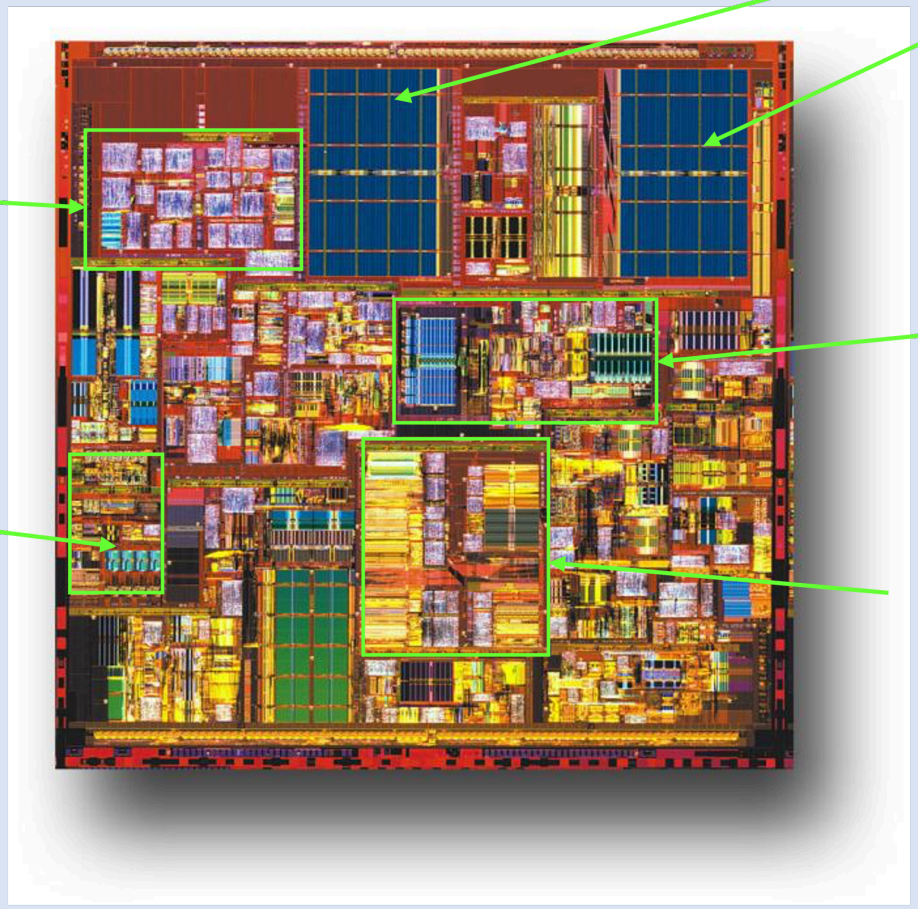
- arithmetic logic unit
- control unit
- clock
- general-purpose registers
- dedicated registers, including:
  - program counter
  - current instruction register
  - memory address register
  - memory buffer register
  - status register.

4.7.3.2 The Fetch-Execute cycle and the role of registers within it

Content

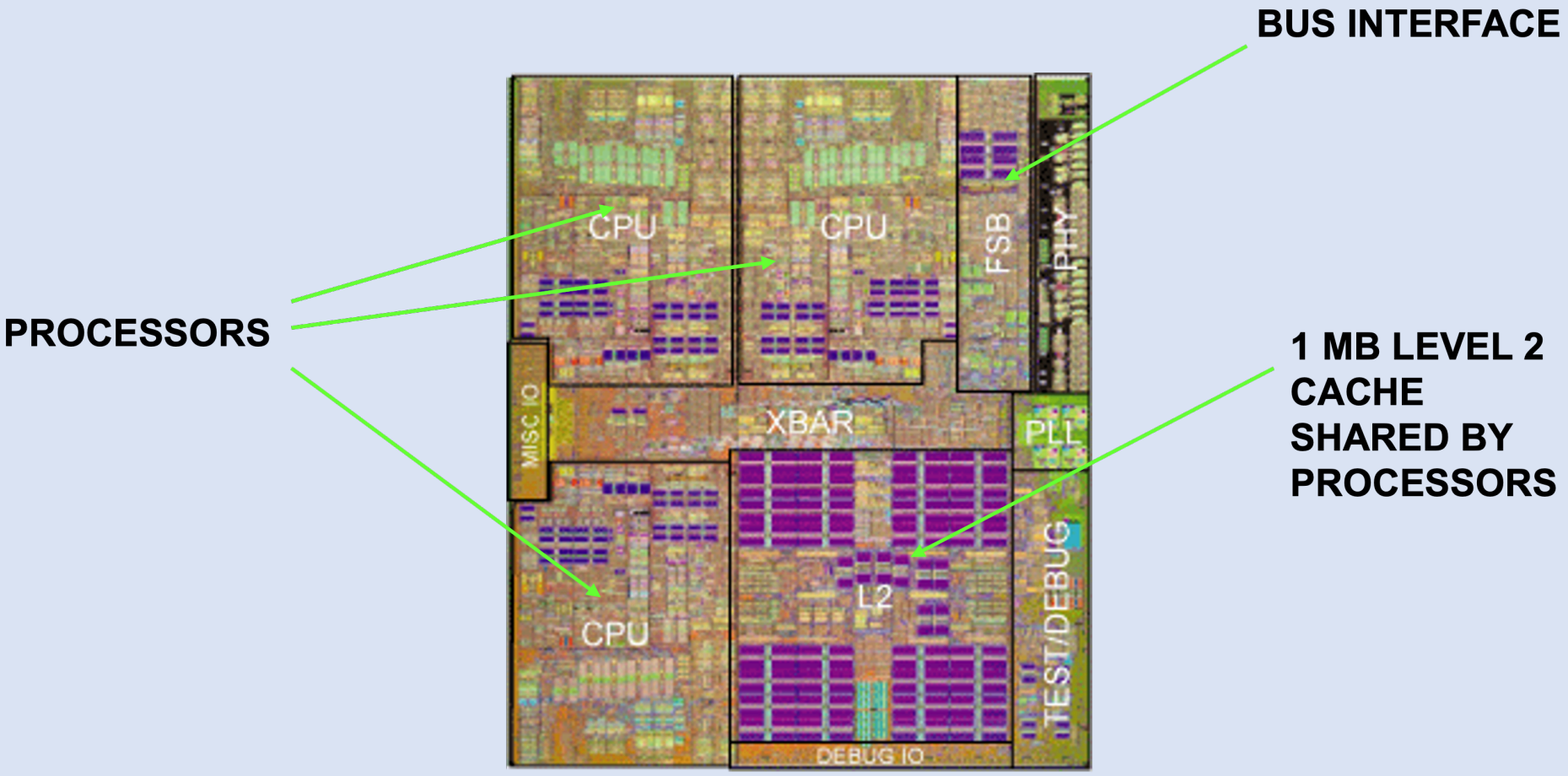
Explain how the Fetch-Execute cycle is used to execute machine code programs including the stages in the cycle (fetch, decode, execute) and details of registers used.

Pentium 4 Processor



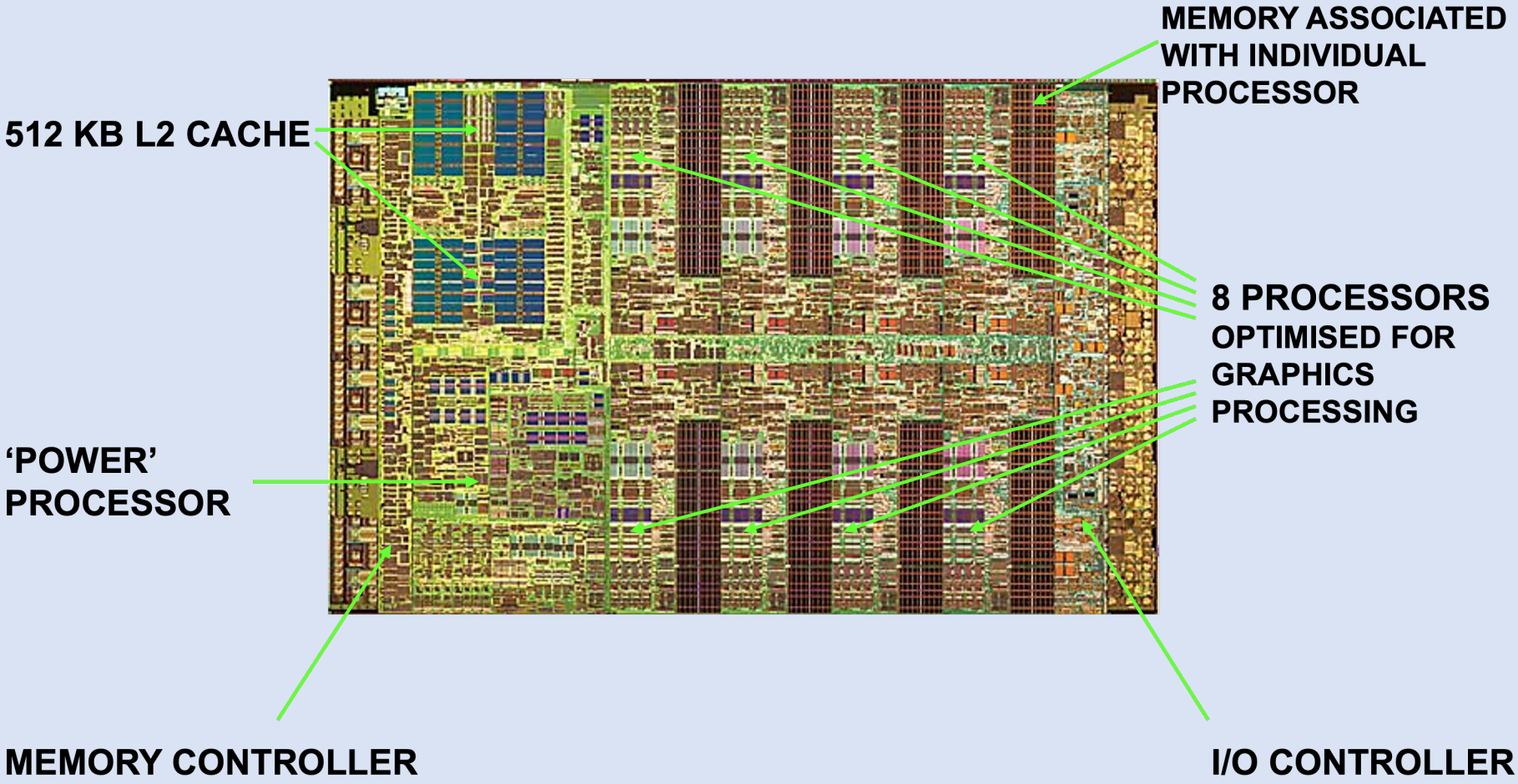
55 million transistors

Xbox 360 Processor



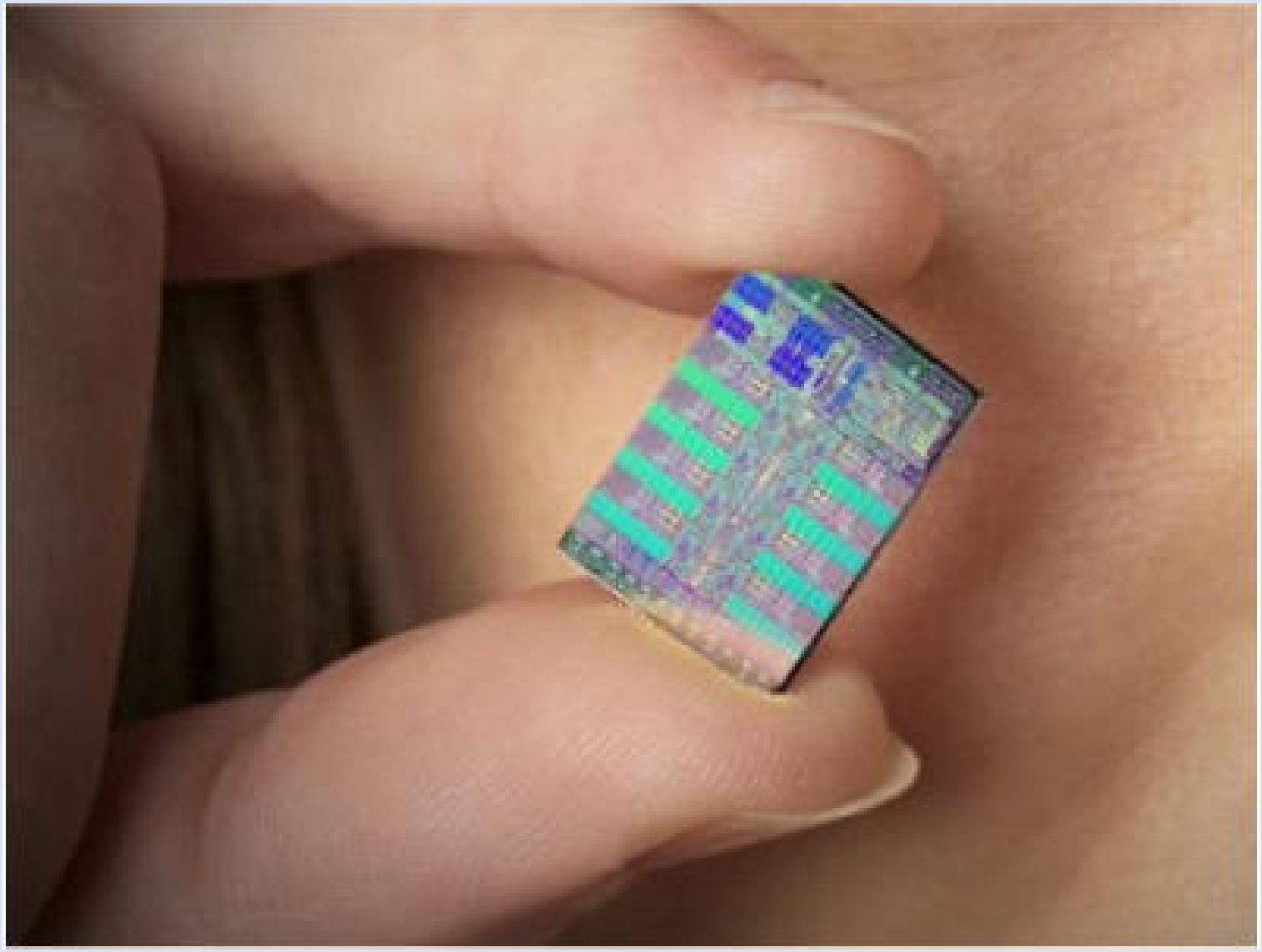
165 million transistors

PlayStation 3 Processor



234 million transistors

PlayStation 3 Processor



Structure of the processor

- **Arithmetic Logic Unit** (ALU)
  - Performs calculations and logic operations (e.g. `ADD`, `OR`, shift)

Structure of the processor

- **Arithmetic Logic Unit** (ALU)
  - Performs calculations and logic operations (e.g. `ADD`, `OR`, shift)
- **Control Unit**
  - Coordinates activities in the CPU, memory and peripherals

Structure of the processor

- **Arithmetic Logic Unit** (ALU)
  - Performs calculations and logic operations (e.g. `ADD`, `OR`, shift)
- **Control Unit**
  - Coordinates activities in the CPU, memory and peripherals
- **Clock**
  - Sends a continuous sequence of clock pulses to step the control unit through its operations

Structure of the processor

- **Arithmetic Logic Unit** (ALU)
  - Performs calculations and logic operations (e.g. `ADD`, `OR`, shift)
- **Control Unit**
  - Coordinates activities in the CPU, memory and peripherals
- **Clock**
  - Sends a continuous sequence of clock pulses to step the control unit through its operations
- **Registers**
  - Very fast on-chip memory for temporary storage of binary values

Structure of the processor

- **Arithmetic Logic Unit** (ALU)
  - Performs calculations and logic operations (e.g. `ADD`, `OR`, shift)
- **Control Unit**
  - Coordinates activities in the CPU, memory and peripherals
- **Clock**
  - Sends a continuous sequence of clock pulses to step the control unit through its operations
- **Registers**
  - Very fast on-chip memory for temporary storage of binary values
- **Internal buses & logic gates**

The role of the control unit

To marshal / control operation of fetch-execute cycle;
Controls fetching / loading / storing operations; **N.E.** fetches instructions
Determines the type of an instruction; **A.** decodes instructions
To execute (some) instructions;
To synchronise operation of processor;
To send control signals / commands to other components;
To control the transfer of data between registers;
To handle interrupts;

This is the mark scheme for a past exam question:

Describe the role of the control unit

Recall

What is the stored program concept?

Recall

What is the stored program concept?

- A program’s instructions must be resident in main memory for it to be executed

Recall

What is the stored program concept?

- A program's instructions must be resident in main memory for it to be executed
- Instructions are fetched and executed one at a time by the processor

Let's see how that second point happens...

Fetch-decode-execute cycle (GCSE)

- **Fetch**: The next instruction is copied from main memory into the processor

Fetch-decode-execute cycle (GCSE)

- **Fetch**: The next instruction is copied from main memory into the processor
- **Decode**: The instruction is interpreted/decoded based on the processor's instruction set

Fetch-decode-execute cycle (GCSE)

- **Fetch**: The next instruction is copied from main memory into the processor
- **Decode**: The instruction is interpreted/decoded based on the processor's instruction set
- **Execute**: The instruction gets executed (carried out). This may involve accessing main memory for `LOAD` and `STORE` instructions, or using the ALU for maths and logic instructions

This sequence is repeated for each instruction.

Processor instructions

- Instructions do not look like the code we have been writing so far (e.g. C#)
- Each instruction consists of an **opcode** and zero or more **operands**
- They are represented in a binary format known as **machine code**
- We can represent binary instructions in **assembly** to improve human readability

Processor instructions

- Instructions do not look like the code we have been writing so far (e.g. C#)
- Each instruction consists of an **opcode** and zero or more **operands**
- They are represented in a binary format known as **machine code**
- We can represent binary instructions in **assembly** to improve human readability

```
0001 0001 0000 1111
```

Processor instructions

- Instructions do not look like the code we have been writing so far (e.g. C#)
- Each instruction consists of an **opcode** and zero or more **operands**
- They are represented in a binary format known as **machine code**
- We can represent binary instructions in **assembly** to improve human readability

```
0001 0001 0000 1111          ADD R1, #15
```

Processor instructions

- Instructions do not look like the code we have been writing so far (e.g. C#)
- Each instruction consists of an **opcode** and zero or more **operands**
- They are represented in a binary format known as **machine code**
- We can represent binary instructions in **assembly** to improve human readability

```
0001 0001 0000 1111
```

```
ADD R1, #15
```

- `ADD` is the opcode
- `R1` and `#15` are operands

Instruction sets

- Each processor has its own set of instructions that it can decode and execute
- This is known as an **instruction set**

Instruction sets

- Each processor has its own set of instructions that it can decode and execute
- This is known as an **instruction set**
- There are some common instruction set architectures that many processors are based on

Instruction sets

- Each processor has its own set of instructions that it can decode and execute
- This is known as an **instruction set**
- There are some common instruction set architectures that many processors are based on
  - Most Windows machines are based on the x86 instruction set architecture
    - Intel
    - AMD

Instruction sets

- Each processor has its own set of instructions that it can decode and execute
- This is known as an **instruction set**
- There are some common instruction set architectures that many processors are based on
  - Most Windows machines are based on the x86 instruction set architecture
    - Intel
    - AMD
  - ARM is on the rise – it has excellent power efficiency
    - Raspberry Pi
    - Apple Silicon (M1-M4, A4-A18 etc.)
    - Snapdragon

Registers

- Small blocks of memory on the processor itself
  - Incredibly fast, because it’s directly on the processor
  - Cannot be much, because the processor is too small

Registers

- Small blocks of memory on the processor itself
  - Incredibly fast, because it's directly on the processor
  - Cannot be much, because the processor is too small
- Some are general-purpose
  - Used within machine code for temporary storage
- Some are dedicated registers, used for the operation of the computer itself

MAR & MBR

- **Memory address register (MAR)**
  - Holds the **address** of a memory location from which data will be read from or written to
  - Processor's direct connection to the **address bus** for accessing main memory
  - Can hold the address of the next instruction to be fetched, or the address of data to be used in an instruction

MAR & MBR

- **Memory address register (MAR)**
  - Holds the **address** of a memory location from which data will be read from or written to
  - Processor's direct connection to the **address bus** for accessing main memory
  - Can hold the address of the next instruction to be fetched, or the address of data to be used in an instruction
- **Memory buffer register (MBR)**
  - Temporarily stores **data** read from or to be written to memory
  - Processor's direct connection to the **data bus** for accessing main memory
  - Can hold the next instruction to be executed or data to be used in an instruction

Program counter (PC)

- Holds the address of the next instruction to be executed
- Incremented as part of the fetch-decode-execute cycle

Current instruction register (CIR)

- Stores the current instruction in binary
- The “decode” step is performed on the CIR

Status register (SR)

- Each bit corresponds to a particular status value
  - Overflow
  - Negative result
  - Zero result
  - ...
- Each bit is set (`1`) or cleared (`0`) depending on the result of the operation

Fetch-decode-execute cycle (GCSE)

- **Fetch**: The next instruction is copied from main memory into the processor

Fetch-decode-execute cycle (GCSE)

- **Fetch**: The next instruction is copied from main memory into the processor
- **Decode**: The instruction is interpreted/decoded based on the processor's instruction set

Fetch-decode-execute cycle (GCSE)

- **Fetch**: The next instruction is copied from main memory into the processor
- **Decode**: The instruction is interpreted/decoded based on the processor's instruction set
- **Execute**: The instruction gets executed (carried out). This may involve accessing main memory for `LOAD` and `STORE` instructions, or using the ALU for maths and logic instructions

This sequence is repeated for each instruction.

Fetch-decode-execute cycle (A-level)

- **Fetch**: The next instruction is copied from main memory into the processor

Fetch-decode-execute cycle (A-level)

- **Fetch**: The next instruction is copied from main memory into the processor
  1. The contents of the **PC** are copied into the **MAR**

Fetch-decode-execute cycle (A-level)

- **Fetch**: The next instruction is copied from main memory into the processor
  1. The contents of the **PC** are copied into the **MAR**
  2. The **address bus** is used to transfer this address to **main memory**

Fetch-decode-execute cycle (A-level)

- **Fetch**: The next instruction is copied from main memory into the processor
  1. The contents of the **PC** are copied into the **MAR**
  2. The **address bus** is used to transfer this address to **main memory**
  3. A read signal is sent along the **control bus**

Fetch-decode-execute cycle (A-level)

- **Fetch**: The next instruction is copied from main memory into the processor
  1. The contents of the **PC** are copied into the **MAR**
  2. The **address bus** is used to transfer this address to **main memory**
  3. A read signal is sent along the **control bus**
  4. The instruction held at that address in memory is transferred via the **data bus** to the **MBR**

Fetch-decode-execute cycle (A-level)

- **Fetch**: The next instruction is copied from main memory into the processor
  1. The contents of the **PC** are copied into the **MAR**
  2. The **address bus** is used to transfer this address to **main memory**
  3. A read signal is sent along the **control bus**
  4. The instruction held at that address in memory is transferred via the **data bus** to the **MBR**
  5. The **PC** is incremented to hold the address of the next instruction to be executed

Fetch-decode-execute cycle (A-level)

- **Fetch**: The next instruction is copied from main memory into the processor
  1. The contents of the **PC** are copied into the **MAR**
  2. The **address bus** is used to transfer this address to **main memory**
  3. A read signal is sent along the **control bus**
  4. The instruction held at that address in memory is transferred via the **data bus** to the **MBR**
  5. The **PC** is incremented to hold the address of the next instruction to be executed
  6. The contents of the **MBR** are copied into the **CIR**

Fetch-decode-execute cycle (A-level)

- **Fetch**: The next instruction is copied from main memory into the processor
  1. The contents of the **PC** are copied into the **MAR**
  2. The **address bus** is used to transfer this address to **main memory**
  3. A read signal is sent along the **control bus**
  4. The instruction held at that address in memory is transferred via the **data bus** to the **MBR**
  5. The **PC** is incremented to hold the address of the next instruction to be executed
  6. The contents of the **MBR** are copied into the **CIR**

Fetch-decode-execute cycle (A-level)

- **Decode**: The instruction gets interpreted/decoded based on the processor's instruction set

Fetch-decode-execute cycle (A-level)

- **Decode**: The instruction gets interpreted/decoded based on the processor's instruction set
  7. The instruction held in the **CIR** is decode by the **control unit** into **opcode** and **operands**

Fetch-decode-execute cycle (A-level)

- **Execute**: The instruction gets executed (carried out)

Fetch-decode-execute cycle (A-level)

- **Execute**: The instruction gets executed (carried out)
  8. The **ALU** is used for maths and logic instructions

Fetch-decode-execute cycle (A-level)

- **Execute**: The instruction gets executed (carried out)
  8. The **ALU** is used for maths and logic instructions
  9. For `LOAD` and `STORE` instructions, the **MAR** and **MBR** are used to access data from main memory

Fetch-decode-execute cycle (A-level)

- **Execute**: The instruction gets executed (carried out)
  8. The **ALU** is used for maths and logic instructions
  9. For `LOAD` and `STORE` instructions, the **MAR** and **MBR** are used to access data from main memory
  10. The **general-purpose registers** and **SR** are updated during this step

Fetch-decode-execute cycle (A-level)

- **Execute**: The instruction gets executed (carried out)
  8. The **ALU** is used for maths and logic instructions
  9. For `LOAD` and `STORE` instructions, the **MAR** and **MBR** are used to access data from main memory
  10. The **general-purpose registers** and **SR** are updated during this step
  11. The **control bus** will transfer signals to other components to initiate or sequence actions

Fetch-decode-execute cycle (A-level)

- **Execute**: The instruction gets executed (carried out)
  8. The **ALU** is used for maths and logic instructions
  9. For `LOAD` and `STORE` instructions, the **MAR** and **MBR** are used to access data from main memory
  10. The **general-purpose registers** and **SR** are updated during this step
  11. The **control bus** will transfer signals to other components to initiate or sequence actions

This sequence is repeated for each instruction.

Ada Quiz: L106 - Processors 1