// Assignment 3

/ The Low-Level Programmer

// by George Hotten

// Polling and Interrupts

// Polling

Polling is where the CPU regularly checks whether a device or peripheral needs attention. This can take up multiple cycles and wastes valuable processing time.

If a device does need attention the CPU ceases what it was currently doing and attends to the needs of the I/O device before continuing executing instructions.

// Interrupts

Interrupts are signals from outside of the CPU that notifies it that a device needs attention, which is checked for at the end of each cycle. This enables the CPU to check for interrupts at the same speed of the CPU's clock.

If an interrupt is found, the CPU will move the contents of the program counter to a temporary location and then load the PC with the addresses from the Interrupt Service Routine. Once the CPU has finished executing the instructions, the previous addresses are moved back into the PC.

// RISC and CISC

// Reduced Instruction Set Computing

A RISC processor contains a smaller and more limited set of instructions. RISC processors are simpler and smaller allowing them to outperform most CISC processors as they can execute instructions within one clock cycle. Code for RISC programmers are often longer but simpler. An example of a RISC CPU is a Qualcomm Snapdragon 888.

/ Advantages

- / Smaller die.
- / Lower TDP.
- / Less heat.
- / Faster performance.
- / Pipelining can be achieved.
- / More cost effective.

/ Disadvantages

- / Performance is dependent on code quality, if instruction scheduling is poor the processor will spend more time waiting than executing code.
- / Lengthy code, RISC processors don't have the luxury of complicated actions within one instruction meaning code becomes very lengthy when writing complex applications.

// Complex Instruction Set Computing

A CISC processor contains a larger and more complex set of instructions. CISC processors are not as efficient as RISC processors are multiple cycles are taken up executing instructions. CISC processors reduce the complexity of programs as complex instructions are housed into a single instruction. An example of a CISC CPU is an AMD Ryzen 5800X.

/ Advantages

/ Simpler programs with less instructions.

/ Easier to program in.

/ Microprogramming is easy to implement.

/ Less complicated compiler.

/ Disadvantages

/ Basic operations can be slow.

/ Calling functions can reduce performance.

/ More expensive to buy.

/ Larger die.

/ Higher TDP.

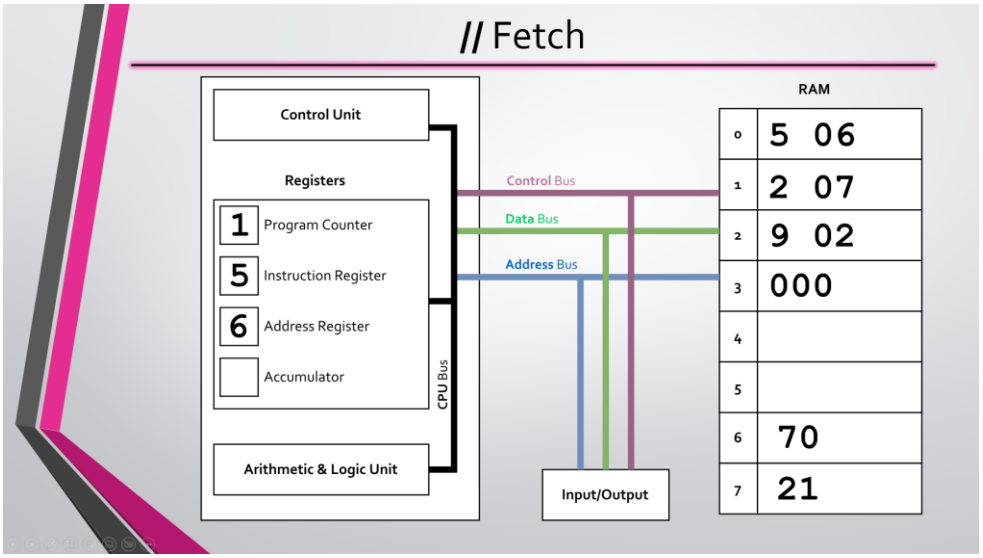
/ Hotter temperatures.

// The Fetch Decode Execute Cycle

The Fetch Decode Execute cycle is the process used by the CPU to retrieve, understand, and execute instructions from the operating system and programs.

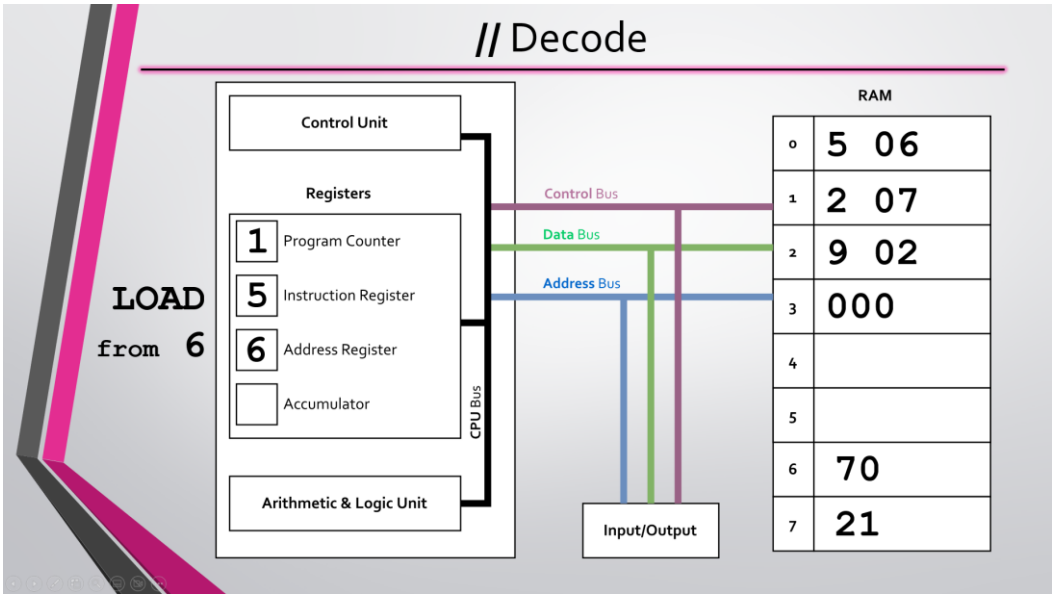
// Fetch

During the fetch cycle, the CPU retrieves the memory location of the next instruction from the program counter. Using the address bus, the CPU requests the contents of the memory location and receives it over the data bus. This instruction is then stored in the Instruction and Address register. The program counter is now incremented by one in preparation for the next cycle.



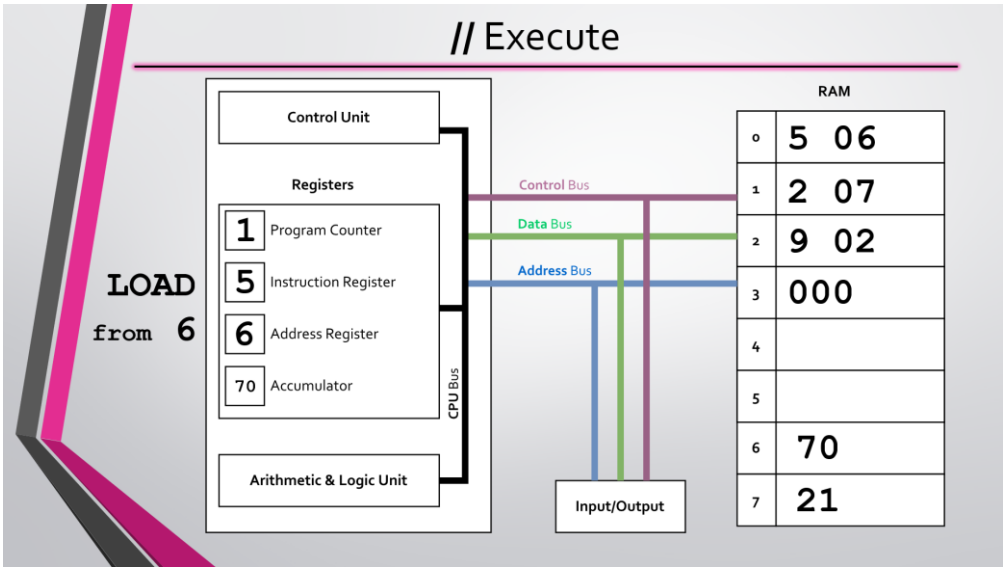
// Decode

After the instructions have been stored in the CPU's registers, the CPU translates the opcode to its corresponding action. For example, if the CPU has 5 in the Instruction register and 6 in the Address register this was translate to LOAD from Address 6.



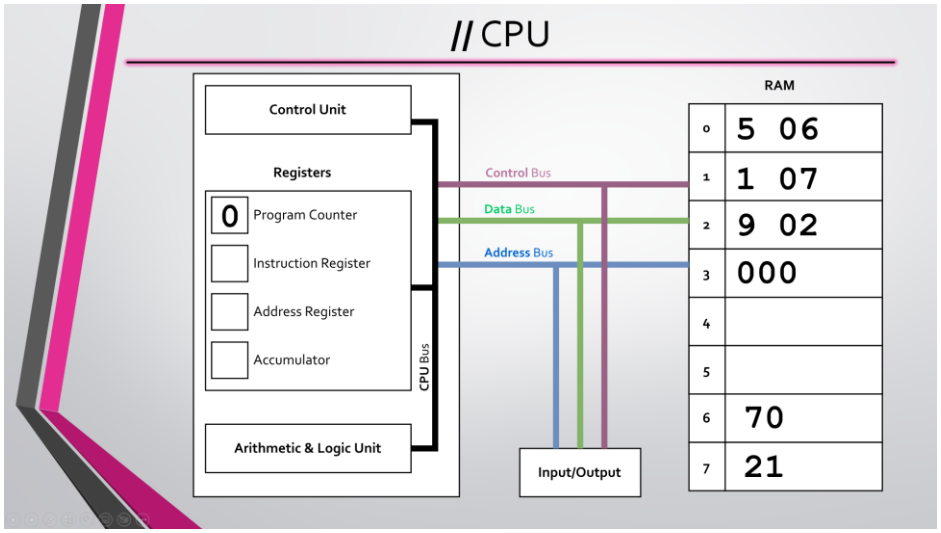
// Execute

Now the CPU knows what to do, it executes the instruction. In this case, loading the data from address 6 into the accumulator.



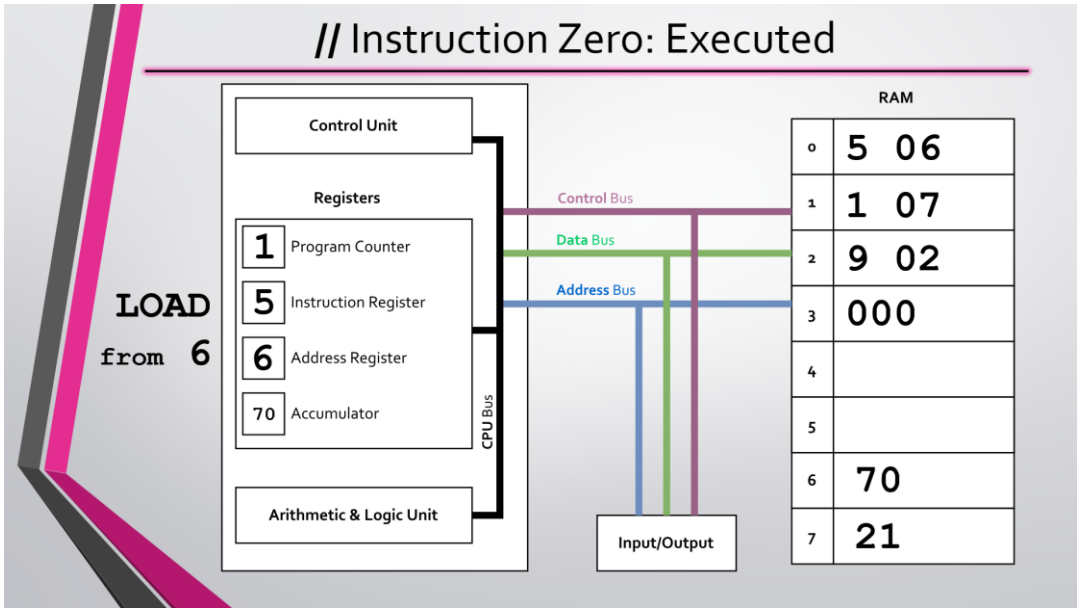
After the execute stage has been finished, the CPU checks to see if an interrupt has been sent. If an interrupt has been sent, the CPU moves the contents of the PC to a separate location and loads the addresses of the interrupt's instructions into the PC. The CPU will execute those instructions using the same steps from above and then resume operations by restoring the previous contents of the PC.

// FDE Example: Adding two numbers together



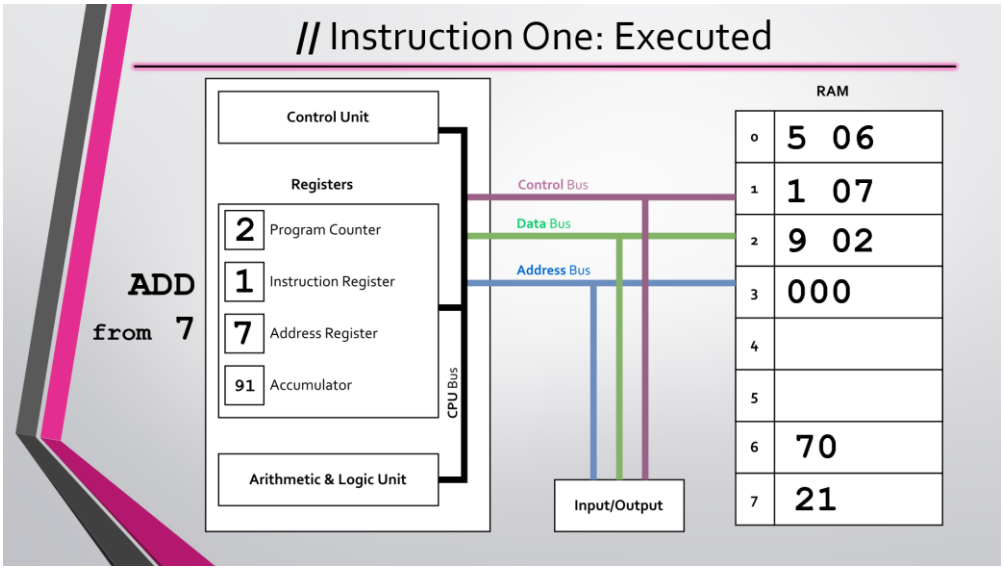
// Instruction Zero: Load number into the accumulator

The CPU fetches and decodes instruction 506 into LOAD from Address 6. This stores 5 into the IR, 6 into the AR and 70 into the accumulator as that is the contents of address 6.



// Instruction One: Add number to the accumulator

Now the CPU fetches and decodes instruction 107 into ADD from Address 7. This stores 1 into the IR, 7 into the AR. The CPU then fetches the 21 from the contents of address 7 and puts it into the ALU along with 70 from the accumulator. The result, 91, is then stored back into the accumulator.



// Instruction Two: Output the result

The CPU now resolves instruction 902 into OUTPUT to the user. This outputs the contents of the accumulator to the user.

// Instruction Three: Halt the program

Finally, the CPU resolves instruction 000 to HALT and ends the program.

// Decision Making and Branching in ASM

```
        INP       ; Ask for user input and store in the accumulator
        STA 70    ; Store the contents of the accumulator in memory address 70
        INP       ; Ask for user input and store in the accumulator
        STA 71    ; Store the contents of the accumulator in memory address 71
        SUB 70    ; Subtract the contents of address 70 from the accumulator
        BRP ylar  ; Jump to function ylar if the accumulator is positive
        LDA 70    ; Load data from address 70 into the accumulator
        OUT       ; Output the accumulator to the user
        HLT       ; Halt the program

ylar    LDA 71    ; Function ylar, load contents of address 71 into the accumulator
        OUT       ; Output the accumulator to the user
        HLT       ; Halt the program
```

This program asks for the user to input two numbers and then displays the largest number. This works by subtracting the first number from the second number, if the number is positive the second number is largest. If the accumulator is negative the first number is largest. This decision is done via the BRP operator which checks if the accumulator is positive. If it is, it jumps to the ylar function.

To see the code working, please review: https://this.is-a-professional-domain.com/8dsxAPw.mp4.

// How the width of system buses affect processor performance and complexity

The address bus and data bus are used to request and receive data from memory respectively. This is important as memory is the primary location where instructions and program data are stored and is what the CPU accesses the most. Therefore, the performance of these buses is vital to the speed of the system.

/ *What is a bus?*

A bus is a set of parallel wires or connectors that are used to transport data between the processor and another component, such as memory or I/O.

/ *What is the width of a bus?*

The width of a bus is the number of bits it can transmit per cycle.

// How does the width of a bus affect performance?

As the width of a bus dictates the number of bits that can be transmitted per cycle, having a larger width allows for more data to be transferred. For example, a 64-bit bus can transmit twice as much data as a 32-bit bus per cycle.

With an increased address bus, more memory can be installed into the system as the CPU can address more locations over the bus. Similarly with an increased data bus, the CPU can receive more complicated instructions and data allowing for each address in memory to store more data freeing up other addresses.

/ Does a large bus width always increase speed?

No, it does not. This is because with the increase of the bus width more data must handled by the CPU per cycle which can slow it down if it was not designed for the larger amount of data. Often, if the speed of the bus is higher a lower bus width doesn't affect the performance as it can transfer data at such a high speed.

// How does a larger bus width affect processor complexity?

The larger the bus width means the more wires and more processing is required by the CPU. For example, a 256-bit bus would require 256 wires going from the CPU to the destination device. This takes up more space on the motherboard and requires the CPU to be able to processes more data per cycle. If the CPU cannot keep up the amount of data being received by the bus it could stall and slow down the CPU.