The Machine Instruction Cycle

Objectives	
2.1.4	Explain the machine instruction cycle.

Before You Begin

The following diagram represents a *procedural abstraction* of the *Machine Instruction Cycle*. This is the process by which a computer system reads and executes commands within its CPU.

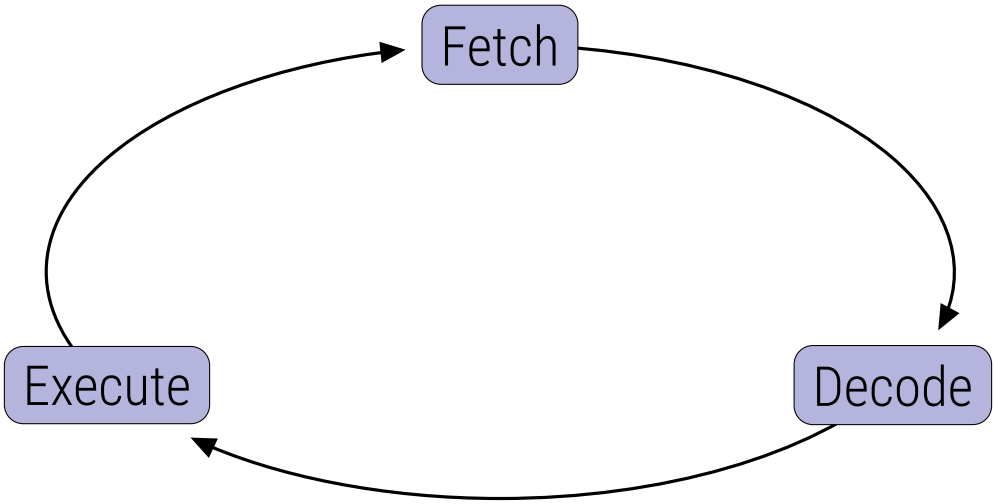

Figure 1: A broad overview of the machine instruction cycle.

Based on your knowledge of the core components of a computer system from our previosu lessons, which part of the CPU do you believe handles each step of the cycle?

Step	Component	Justification
Fetch Instruction		
Decode Instruction		
Execute Instruction		

Question #1 *An abstraction such as the one shown here is known as a* high-level abstraction *because it hides almost all of the complexity of a system in favour of being human readable. What are the benefits and drawbacks to such a high level of abstraction?*

Important Terms

Term	Definition
Current Instruction Register (CIR)	
Instructino	
Interrupt	
Jump	
Program Counter (PC)	

Technical Background

Fetching Instructions

The following flow chart represents a lower-level abstraction of the fetch phase of the instruction cycle.

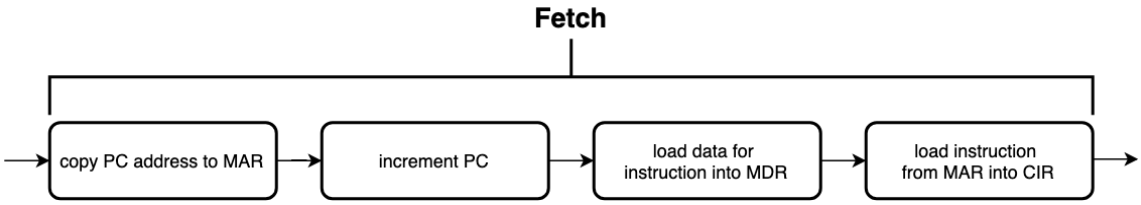

Figure 2: A closer look at the fetch part of the machine instruction cycle.

Notes

Question #2 *The final process in the fetch phase of the machine instruction cycle loads the instruction from the address in one register (MAR) into another (CIR). What benefit to the overall cycle does the use of a second register have?*

The Fetch-Decode-Execute (FDX) Cycle

The following flowchart shows the entire FDX (**F**etch-**D**ecode-E**x**eute) cycle, including special decisions in the case of a jump instruction and the presence of interrupts.

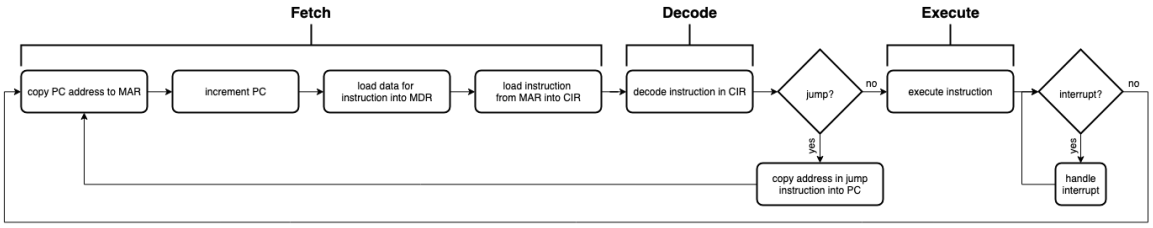

Figure 3: A look at the entire machine instruction cycle.

Notes

Question #3 *Why do you think interrupts are handled as a special step in the machine instruction cycle, rather than simply being added to the instruction queue?*

Sample Instruction Set

The following table shows the first ten instructions available in the popular "x86 Instruction Set" (likely the instruction set being used by your own personal computer's processor).

Note: Repeated instruction names are listed as each operation code ("opcode") refers to instructions manipulating data in different memory locations and registers, which is beyond the scope of this course.

00	01	02	03	04	05	06	07	08	09
ADD						PUSH	POP	OR	

More information about this and related instruction sets are avialalbe through Intel's website: https://software.intel.com/en-us/articles/intel-sdm.

Notes

Question #4 *Although the 8086 processor was first published by Intel in 1974, most of its core instruction set is still in use with more modern Intel and compatible processors. Why is maintaining consistency and "backwards compatibility" despite advances in technology important for the computing and software industries?*

Developing Technical Skills

Assembly Language Programming

Along with the instruction set, the x86 specification also provides the description of an "aseembly language". This family of languages is a very low-level language designed to interact almost directly with the instruction.

In this activity, you'll be looking at a simple assembly language program and adding comments to each line of code. Relevant descriptions of assembly language syntax and commands are given below.

Syntax	Description
ax, bx, etc...	Reference to specific 16-bit data registers provided automatically by the system.
ah, bh, etc... / al, bl, etc...	Reference to the "highest"/"lowest" 8-bits of each data register provided automatically by the system.
ADD A, B	Add the data from B into register A.
A **DW** B	Create a data word named "A" and store the value B.
END MAIN	Denotes the end of the program.
A **ENDP**	Ends the description of a specific procedure named "A".
INT 21H	Call an interrupt to the MS-DOS API interrupt table. Note: the MS-DOS API specifies interrupt "02" as callilng for character output.
MOV A, B	Copy data from B into register A.
A **PROC**	Start the description of a specific procedure named "A".

Add a comment next to each of the following lines of code based on the syntax description above. A few comments have already been made for you.

Note #1: Comments in x86 assembly are separated from the text of the program using a `;` character.
Note #2: Numbers appended with an "H" denote hexadecimal values.

#	Code	Comment
01	`.stack 100H`	`; set aside 256 bits of memory for the program.`
02		
03	`.data`	`; begin the data section of the program`
04	`a DW 02H`	
05	`b DW 03H`	
06		
07	`.code`	`; begin the code section of the program`
08	`MAIN PROC`	
09	`MOV ax, @data`	
10	`MOV ds, ax`	
11		
12	`MOV ax, a`	
13	`MOV bx, b`	
14		
15	`ADD ax, bx`	
16		
17	`MOV ah, 02`	
18	`MOV dx, ax`	
19	`INT 21H`	
20		
21	`MAIN ENDP`	
22	`END MAIN`	

Reflections

Question #5 *In January 2006, Apple began producing a line of their Macintosh computers using the now-standard Intell processor. Previously, they had been using the Apple-IBM-Motorola produced PowerPC line of processors. What impact do you think this switch, along with its change in instruction set, had on software development for Apple computers?*

Question #6 *The processors used by many mobile devices use the ARM architecture, which implements a Reduced Instruction Set Computer (RISC). What benefits and drawbacks does implementing a less complex instruction set in a processor provide?*

Question #7 *Describe at least one new thing you have learned from this lesson. How might you apply this knowledge in the future?*

Question #8 *Select the option which best reflects how confident you are in applying what you have learend in this lesson.*



Question #9 *What further questions do you still have about this lesson's content?*