Assignment 1 - Computer Architecture

Due Feb 8 at 3pm **Points** 25 **Questions** 7
Available Feb 1 at 8:30pm - Feb 8 at 3pm **Time Limit** None
Allowed Attempts Unlimited

Take the Quiz Again

Attempt History

	Attempt	Time	Score
KEPT	Attempt 2	11 minutes	0 out of 25 *
LATEST	Attempt 2	11 minutes	0 out of 25 *
	Attempt 1	2,931 minutes	0 out of 25 *

* Some questions not yet graded

Correct answers are hidden.

Score for this attempt: **0** out of 25 *
Submitted Feb 6 at 10:41pm
This attempt took 11 minutes.

Question 1

Not yet graded / 1 pts

What is the make and model of your primary computer?

Your Answer:

My primary computer is an HP Pavilion 15 Notebook PC with a 15-inch display and the manufacturer of it is Hewlett-Packard. My system's model number is G0X71AV.

Question 2

Not yet graded / 1 pts

What processor is used in your primary computer?

Your Answer:

The processor that is used in my primary computer is Intel(R) Core (TM) i5-4210U CPU @ 1.70GHz, 2401 MHz, 2 Core(s), 4 Logical Processor(s).

Question 3

Not yet graded / 1 pts

Is the processor a RISC or CISC architecture?

Your Answer:

The processor "Intel(R) Core (TM) i5-4210U CPU @ 1.70GHz, 2401 MHz, 2 Core(s), 4 Logical Processor(s)" is based on the Complex Instruction Set Computing (CISC) architecture.

Question 4

Not yet graded / 4 pts

Justify your answer to the previous question based on the characteristics we discussed in class. What makes this processor RISC or CISC?

(Please don't say you looked it up on the Internet a found a reference saying that it's a RISC or CISC!)

Your Answer:

The processor "Intel(R) Core (TM) i5-4210U CPU @ 1.70GHz, 2401 MHz, 2 Core(s), 4 Logical Processor(s)" is based on the Complex Instruction Set Computing (CISC) architecture because it has a large number of complex instructions that can perform multiple operations in a single instruction. This results in more functionality per instruction but also

results in longer instruction length, increased instruction decoding complexity and decreased clock frequency.

CISC architecture has a large number of instructions, each capable of performing multiple tasks, with varying lengths and multiple addressing modes. This leads to higher code density, allowing for more functionality in less code space. The longer instruction length also requires more transistors and a more complex instruction decoder, leading to a slower clock speed.

In contrast, Reduced Instruction Set Computing (RISC) architecture has a smaller and simpler instruction set, with instructions that are all the same length and with a uniform addressing mode. This simplicity allows for higher clock speeds, but requires more instructions to accomplish a task, resulting in lower code density.

Overall, the Intel(R) Core (TM) i5-4210U CPU @ 1.70GHz, 2401 MHz, 2 Core(s), 4 Logical Processor(s) is a CISC processor due to its larger instruction set, with instructions capable of performing multiple operations, and its more complex instruction decoding.

Question 5

Not yet graded / 3 pts

Write a sequence of SIC instructions to swap the contents of two memory locations, *alpha* and *beta*.

Your Answer:

Here is a sequence of SIC (Simple Instruction Set Computer) instructions to swap the contents of two memory locations, alpha and beta:

LDA ALPHA : Load alpha into the accumulator

STA TEMP : Store the contents of the accumulator in a temporary location

LDA BETA : Load beta into the accumulator

STA ALPHA : Store the contents of the accumulator in alpha

LDA TEMP : Load the contents of temp into the accumulator

STA BETA : Store the contents of the accumulator in beta

ALPHA RESW 1 : Reserve 1 word for alpha

BETA RESW 1 : Reserve 1 word for beta

TEMP RESW 1 : Reserve 1 word for temp

This code assumes that 'ALPHA' and 'BETA' are memory locations and 'temp' is a temporary memory location used to store the contents of 'alpha' during the swap. The 'LDA' instruction loads the contents of a memory location into the accumulator and the 'STA' instruction stores the contents of the accumulator in a memory location.

Question 6

Not yet graded / 6 pts

Write a short sequence of SIC instructions:

There are two memory locations, *arrayA* and *arrayB*. Each is the start of an array of one hundred 24-bit words.

Iterate through the two arrays, looking for matching elements (e.g. arrayA[37] == arrayB[37]).

Store the total number of matches in a memory location called *matches*.

Your Answer:

There are two memory locations, arrayA and arrayB. Each is the start of an array of one hundred 24-bit words.

Iterate through the two arrays, looking for matching elements (e.g. arrayA[37] == arrayB[37]).

Store the total number of matches in a memory location called matches.

Here is a short sequence of SIC (Simple Instruction Set Computer) instructions to iterate through two arrays and store the total number of matching elements:

```
LDA ZERO    : Load 0 into the accumulator to initialize the matches count
LDX ZERO    : Load 0 into index register X to initialize the loop counter
LOOP:    : Start of the loop
  LDA ARRAYA, X    : Load the word at the current index from arrayA into the accumulator
  CMP ARRAYB, X    : Compare the word at the current index from arrayA with the word from arrayB
  JEQ MATCH    : Jump to MATCH if the words are equal
  JLT ENDLOOP    : Jump to ENDLOOP if X is equal to 100
  INX    : Increment the loop counter
  JMP LOOP    : Jump to the start of the loop
MATCH:    : Start of the match routine
  INR MATCHES    : Increment the matches count
  JMP LOOP    : Jump to the start of the loop
ENDLOOP:    : End of the loop
  STA MATCHES    : Store the matches count in the MATCHES memory location
```

This code assumes that 'ARRAYA', 'ARRAYB', 'ZERO', and 'MATCHES' are memory locations, 'X' is the index register, 'INR' increments the contents of a memory location, and 'INX' increments the contents of the index register. The loop iterates 100 times and the 'JEQ' instruction jumps to the 'MATCH' routine if the words at the current index of 'ARRAYA' and 'ARRAYB' are equal. The 'JLT' instruction jumps to the 'ENDLOOP' routine if 'X' is equal to 100, signaling the end of the loop. The 'STA' instruction stores the contents of the accumulator in the 'MATCHES' memory location, representing the total number of matching elements.

Question 7

Not yet graded / 9 pts

Write a short sequence of SIC/XE instructions:

There are three memory locations, *arrayA*, *arrayB,* and *arrayC*. Each is the start of an array of one hundred 48-bit floating point numbers.

Iterate through *arrayA* and *arrayB.* Multiply each floating point number in *arrayA* by the corresponding floating point number in *arrayB*, and place

the result in arrayC. (E.g. arrayC[37] == arrayA[37] * arrayB[37]).

Your Answer:

Here is a short sequence of SIC/XE (Simple Instruction Set Extended Computer) instructions to iterate through two arrays, multiply the corresponding elements, and store the results in a third array:

```
LDX ZERO    : Load 0 into index register X to initialize the loop counter
LOOP:    : Start of the loop
  LDF ARRAYA, X    : Load the floating point number at the current index from arrayA into the floating point accumulator
  MULF ARRAYB, X    : Multiply the floating point number at the current index from arrayB with the accumulator
  STF ARRAYC, X    : Store the result in arrayC at the current index
  INX    : Increment the loop counter
  CMPX #100    : Compare the loop counter with 100
  JLT LOOP    : Jump to the start of the loop if X is less than 100
```

This code assumes that 'ARRAYA', 'ARRAYB', 'ARRAYC', 'ZERO' are memory locations, 'X'is the index register, 'LDF' loads a floating point number from memory into the floating point accumulator, 'MULF' multiplies the contents of the floating point accumulator with a floating point number from memory, 'STF' stores the contents of the floating point accumulator in memory, and 'CMPX' compares the contents of the index register with an immediate value. The loop iterates 100 times and the 'JLT' instruction jumps to the start of the loop if 'X' is less than 100, signaling that the loop should continue. The 'MULF' instruction multiplies the contents of the floating point accumulator with the floating point number at the current index of 'ARRAYB', and the 'STF' instruction stores the result in 'ARRAYC' at the current index.

Quiz Score: **0** out of 25