|  |  |  |  |
| --- | --- | --- | --- |
| Instructor | ***Luke Papademas*** | Due Date | **6/30** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Part | **1** | **2** | **3** | **4** | Total |
| *Maximum Points* | **25** points | **25** points | **25** points | **25** points | **100**G101010 pointsG |
| ***Your Score*** |  |  |  |  |  |

**Textbook Reading Assignment**

Thoroughly read Chapter(s) 5 in your Computer Architecture and Organization textbook.

**Part 1 Glossary Terms - Instruction Set Architectures ( ISAs )**

Define, in detail, each of these glossary terms from the realm of computer architecture and computer topics, in general. If applicable, use examples to support your definitions. Consult your notes

or course textbook(s) as references or the Internet by visiting Web sites such as:

[**http://www.ask.com**](http://www.ask.com) or [**http://www.webopedia.com**](http://www.webopedia.com/)

**(a) endian**

|  |
| --- |
| Endian refers to the computer architecture’s byte order. There are two types – little endian, which means the least significant byte is stored at the lowest memory address, and big endian, which means that the most significant byte is stored at the lowest memory address. |

**(b) infix Notation**

|  |
| --- |
| Arithmetic notation that places the operator between the operands. |

**(c) instruction - level pipelining**

|  |
| --- |
| Instruction level pipelining is the process of the CPU breaking down the fetch-decode-execute cycle into smaller steps and running these steps in parallel. Each step is called a pipeline stage, and stages are combined to form a pipe. This allows the fetch-decode-execute cycle to balance the time taken by each pipeline stage. |

**(d) instruction set**

|  |
| --- |
| An instruction set is the set of commands that instruct the computer with what to do. The instruction set includes commands for things such as data movement, arithmetic, Boolean logic, bit manipulation, I/O, transfer of control, and other special purpose commands. |

**(e) postfix Notation**

|  |
| --- |
| Arithmetic notation that places the operator after the operands. |

**Part 2 Exercises - Instruction Set Architectures ( ISAs )**

For each of the following, enter True or False.

\_\_\_T\_\_ **(1)** Most computers typically fall into one of three types of CPU organization:

(1) general register organization; (2) single accumulator organization; or (3) stack organization.

\_\_F\_\_\_ **(2)** The advantage of zero - address instruction computers is that they have short programs; the disadvantage is that the instructions require many bits, making them very long.

\_\_T\_\_\_ **(3)** An instruction takes less time to execute on a processor using an instruction pipeline than on a processor without an instruction pipeline.

\_\_T\_\_\_ **(4)** The term " endian " refers to an architecture’s byte ordering.

\_\_T\_\_\_ **(5)** Stack architectures have good code density and a simple model for evaluation of expressions, but do not allow random access, which can cause a problem with the generation of efficient code.

\_\_F\_\_\_ **(6)** Most architectures today are accumulator - based.

\_\_T\_\_\_ **(7)** Fixed - length instruction format typically results in better performance than variable length instruction format.

\_\_F\_\_\_ **(8)** Expanding opcodes make instruction decoding much easier than when it is not used.

\_\_F\_\_\_ **(9)** Instruction set orthogonality refers to the characteristic in an instruction set architecture where each instruction has a " backup " instruction that performs the same operation.

\_\_T\_\_\_ **(10)** The effective address of an operand is the value of its actual address in memory.

**Part 3 Exercises - Reverse Polish Notation ( RPN )**

Convert each of the following and use an online calculator, such as that shown below, to check your answers.

[**https://www.mathblog.dk/tools/infix-postfix-converter/**](https://www.mathblog.dk/tools/infix-postfix-converter/)

**(1)** Convert the following expression from infix to Reverse Polish ( postfix ) Notation.

( 8 − 6 ) / 2 8 6 2 - / = 8 6 - 2 8 6 2 / / -

**(2)** Convert the following expression from infix to Reverse Polish ( postfix ) Notation.

( 2 + 3 ) × 8 / 10 = 23 + 8 x 10 /

**(3)** Convert the following expression from infix to Reverse Polish ( postfix ) Notation.

5 × ( 4 + 3 ) × 2 − = 5 4 3 + \* 2 \* -

**(4)** Convert the following expressions from infix to Reverse Polish ( postfix ) Notation.

X × Y + W × Z + V × U = X Y x W Z x + V U x +

**(5)** Convert the following expressions from Reverse Polish Notation to infix notation.

W X Y Z − + × = W X Y Z − \* +

**Part 4 Exercises - Instruction Set Architectures ( ISAs )**

Write a complete answer for each of these. You can use an online converter such as that available at the following Web address.

[**http://scanftree.com/Data\_Structure/prefix-postfix-infix-online-converter**](http://scanftree.com/Data_Structure/prefix-postfix-infix-online-converter)

**(1)** Is the following a valid postfix operation? If so, explain how a stack is used to evaluate the RPN expression or explain why it is invalid.

12 8 3 1 + − /

|  |
| --- |
| **This is a valid expression. The stack pushes the 12, 8, 3, and 1 onto the stack. Then, it sees an addition symbol and pops the top two values off of the stack (1 and 3), performs addition, and pushes 4 back onto the stack. Then, it sees a subtraction symbol and pops 4 and 8 off the stack, and performs 8 – 4, and pushes 4 back onto the stack. Then is sees a division symbol and does 12 / 4 and pushes 3 back onto the stack.** |

**(2)** Determine the arithmetic statement for which the following program was written to evaluate the statement using a stack organized computer with zero - address instructions ( so only pop and push can access memory ) .

Push A

Push B

Subtract

Push C

Push D

Push E

Mult

Push F

Subtract

Mult

Add

Push G

Push H

Push K

Mult

Add

Div

Pop X

|  |
| --- |
| **X = ((A – B) + C (D x E - F)) / (G + (H x K))** |

**(3)** What is the difference between using direct and indirect addressing? Give an example.

|  |
| --- |
| **Direct addressing is when the value to be referenced is obtained by specifying its memory address directly in the instruction. For example, if the instruction is Load 008, the value found at memory address 008 is loaded. Indirect addressing is when the bits in the address field specify a memory address to be used as a pointer. The effective address of the value is found by going to the memory address provided in the instruction. For example, if the instruction is Load 008, and the value 0x2A0 is found in location 008, then the value we are looking for is at address 0x2A0.** |