FCS 2450 Term Project – Milestone One

Introduction

Utah Valley University (UVU) hired your company IED (Innovative EDucation) to develop a software simulator called UVSim for computer science students to learn machine language and computer architecture. Students can execute their machine language programs on the simulator.

The UVSim is a simple virtual machine, but powerful. The UVSim can only interpret a machine language called BasicML.

The UVSim contains CPU, register, and main memory. An accumulator – a register into which information is put before the UVSim uses it in calculations or examines it in various ways. All the information in the UVSim is handled in terms of words. A word is a signed four-digit decimal number, such as +1234, -5678. The UVSim is equipped with a 100-word memory, and these words are referenced by their location numbers 00, 01, …, 99. The BasicML program must be loaded into the main memory starting at location 00 before executing.

Each instruction written in BasicML occupies one word of the UVSim memory (instruction are signed four-digit decimal number). We shall assume that the sign of a BasicML instruction is always plus, but the sign of a data word may be either plus or minus. Each location in the UVSim memory may contain an instruction, a data value used by a program or an unused area of memory. The first two digits of each BasicML instruction are the operation code specifying the operation to be performed.

BasicML vocabulary defined as follow.

I/O operation:

READ = 10	Read a word from the keyboard into a specific location in memory.
WRITE = 11	Write a word from a specific location in memory to screen.

Load/store operations:

LOAD = 20	Load a word from a specific location in memory into the accumulator.
STORE = 21	Store a word from the accumulator into a specific location in memory.

Arithmetic operation:

Add = 30	Add a word from a specific location in memory to the word in the accumulator (leave the result in the accumulator)
SUBTRACT = 31	Subtract a word from a specific location in memory from the word in the accumulator (leave the result in the accumulator)

DIVIDE = 32	Divide the word in the accumulator by a word from a specific location in memory (leave the result in the accumulator).
MULTIPLY = 33	multiply a word from a specific location in memory to the word in the accumulator (leave the result in the accumulator).

Control operation:

BRANCH = 40	Branch to a specific location in memory
BRANCHNEG = 41	Branch to a specific location in memory if the accumulator is negative.
BRANCHZERO = 42	Branch to a specific location in memory if the accumulator is zero.
HALT = 43	Pause the program

The last two digits of a BasicML instruction are the operand – the address of the memory location containing the word to which the operation applies.

Deliverable

(1) (5 points) Meeting log. Every team member need to sign contribution form. Submission without signature will be count as late, and late submission rules will be applied.

(2) (10 points) Decompose the milestone into backlogs and sprints.

(3) (10 points) (Hard Copy) Write one page to compare SCRUM and one another process model (single line space and 12 pt size font).

(a) Give 3 reasons to support, and 3 reasons to against SCRUM process model.

(b) Give 3 reasons to support, and 3 reasons to against the process model your team chose.

(4) (20 points) (Hard Copy) Describe the system to all stakeholders by using Use Case Diagram. The number of use cases is no more than 15 and no less than 10. The range of number is used to control the granularity of the use cases. Try your best to make sure the use cases are in the same detailed level.

(5) (40 Points) Develop a prototype of UVSim. The prototype should be console application and run from commend line. You are free to use any object oriented language. Every team member need to make comment on your part of the implementation.

(6) (10 points) You need to test all the function of your prototype to show reasonable quality to release. Design your test cases. Each test case for each function. You should have 10 to 15 test cases. Each test case should have inputs and expected outputs.

(7) (5 points) Write Readme to instruct user how to use your product and what is the running environment. The grader should be able to run your program follow your read me. If the grader

could not run your program following your Readme, that means your Readme is not clear and will not be awarded credits.