CS 474: Object Oriented Programming Languages and Environments
Fall 2014

Second Smallalk project

Due time: **9:00 pm on Monday 10/27/2014**

Instructor: Ugo Buy
TA: Bowen Dong

You are required to implement an *Assembly Language Emulator (ALE)* in Cincom Smalltalk. The assembly language is called *Simple Assembly Language (SAL)*. It only has a very limited set of instructions. The ALE keeps three kinds of information in appropriate widgets:

1. An editor field for entering and editing assembly language source code (one instruction per line).
2. An *action button* for executing one line of code, starting from line 0. (See the description of the PC below.)
3. An *action button* for executing the entire program, starting from the current value of the PC. (See below.)
4. A set of widgets for displaying the state of the computer (memory and registers) and state changes brought about by the execution of SAL instructions.

The computer hardware uses 16-bit words and consists of the following components:

1. *Data memory.* A 16-bit, word-addressable memory (RAM) for data, holding 64 words. Words are addressed by their location, starting from location 0 all the way up to location 63. Each location holds a signed integer in 2's complement notation.
2. *Program memory.* This stores the SAL program being executed. It can store up to 256 instructions, numbered from 0 to 255.
3. *Accumulator.* A 16-bit register. It is also known as *Register A* or *A* for short.
4. *Extra register.* A 16-bit register also known as *Register B* or *B* for short.
5. *Program counter (PC).* An 8-bit program counter (PC). The PC holds the address (number in program memory) of the next instruction to be executed. Before the program starts execution, the PC holds the value 0. It is subsequently updated as each instruction is executed.
6. A *zero-result bit.* This bit is set if the last ADD instruction produced a zero result. This bit is cleared if the last ADD instruction produced a result different from zero. The initial value is zero. The bit is changed only after ADD instructions are executed.
7. An *overflow bit.* This bit is set whenever an ADD instruction produces an overflow (i.e., a result that cannot be stored in 2's complement notation with 16 bits). It is cleared if the ADD instruction did not produce an overflow. The initial value is zero.

The registers are used to hold data in arithmetic operations (i.e., additions). The program counter holds the index value (starting at 0) of the next instruction to be executed.

SAL has the following instruction set:

You may assume that SAL programs are entered correctly by users of ALE. (You are not required to perform error diagnosis or correction). Information in data memory and the registers can be displayed

LDA *number*	Loads byte at data memory address *number* into the accumulator.
LDB *number*	Loads byte at data memory address *number* into *B*.
LDI *value*	Loads the integer *value* into the accumulator register. The value could be negative but must be in the ran
ST *number*	Stores content of accumulator into data memory at address *number*.
XCH	Exchanges the content registers *A* and *B*.
JMP *number*	Transfers control to instruction at address *number* in program memory.
JZS *number*	Transfers control to instruction at address *number* if the zero-result bit is set.
JVS *number*	Transfers control to instruction at address *number* if the overflow bit is set.
ADD	Adds the content of registers *A* and *B*. The sum is stored in *A*. The overflow and zero-result bits are set or cleared as needed.
HLT	Terminates program execution.

Table 1: Instruction set of SAL.

in binary, decimal, or hexadecimal format. Program source code should be displayed in symbolic (i.e., textual) format. Finally, you must use inheritance in your implementation. One good place for inheritance is to define an abstract superclass for SAL instructions with concrete subclasses for each particular instruction. Beware of infinite loops in SAL.

You must work alone on this project. Your project code should be in a special package called CS474. Save all your code by filing out that package in the file xxx.st, where xxx denotes your last name. Submit the file by clicking on the link provided with this assignment. No late submissions will be accepted.