Mobile processor Programming assignment #1: Simple calculator

due by March 23, 2018

Introduction

Computer performs ‘computation’ according to given instructions. The computation can be either arithmetic operations, logical operations, or remembering/storing information. Throughout the semester, we will make a simple computer in software that performs such computations with a computer language.

To begin with, we will make a simple calculator to understand the operations beneath the computers. The simple calculator can do the following binary operations (two operands) such as plus, minus, multiply, and divide.

The calculator needs ‘input instruction’ to execute with. By default, the calculator uses ‘input.txt’ file as input file. From the file, the calculator reads in the instructions, and executes the read-in instruction one by one.

More specific requirements are given as follows:

1. The calculator supports the following operations: addition, subtraction, multiplication, division.
    A. Each operation begins with operator symbols, such as ‘+’, ‘-‘, ‘*’, ‘/’, and ends with newline. That is, an instruction is written in a line. The operands are followed by the operator. The operands are separated by white spaces.
2. Then the calculator decodes the instruction in order to identify the operands (required resources).
    A. An operand can be either an intermediate value (normal hexa-decimal number) or a register number.
    B. To distinguish the operand types, operand has a prefix. The intermediate value has prefix ‘0x’, and the register number has prefix ‘R.’
3. The calculated result is stored in the register number 0, or R0
4. Calculator supports the register move operations: M

A. M operation supports intermediate value-to-register move. That is, M can be defined with one intermediate value and one register number. The M operation moves intermediate value to the designated register. Intermediate value has prefix '0x', and register number has prefix 'R'.

B. M operation supports register-to-register move. The M operation can take two registers Rs and Rd. 'M Rs Rd' moves value of source register, Rs to destination register, Rd. The register number has prefix 'R', similarly to the previous rule.

5. The calculator prints out the computation results at every step.

   A. The calculator has to print out what it does in a complete in-order form.

   B. Examples) input → output

      i. + 0x3 0x2 → R0: 5 = 3+2

      ii. – 0x1 0x2 → R0: -1 = 1 – 2

      iii. M R2 0x3 → R2: 3

      iv. + R0 R2 → R0: 2 = -1 + 3

6. Handle exception cases gracefully.

7. To perform calculation, the calculator has to fetch the instruction from the file; calculator should define a new variable 'inst_reg', which stores the instruction string.

   A. The calculator has an instruction pointer that points to the memory address of the executing instruction.

No restrictions on language. No copy allowed. Put your code on github so that the other students can check it.

Optional implementation (Extra)

1. You can compare two register values with 'C' instruction that compares the followed two operands. Then, the calculator saves the result in R0. The result value is zero, -1, or +1. When the two operands are the same, then the result is zero. When the first operand is larger than the second one, then the result is +1; otherwise, -1.

2. You can define some additional ‘J’ instruction. ‘J’ is a jump instruction that jumps to somewhere the second operand points to. The ‘J’ instruction takes single operand, and the operand is the number of input instruction in the input file.
3. Another interesting instruction is ‘BEQ.’ BEQ is a conditional jump instruction. The instruction compares r0 with zero. If R0 is zero, then the calculator jumps to the target instruction. Otherwise, execute the next instruction.
4. You can make some input program such as calculating GCD (greatest common divisor) of two numbers. With the given instruction, try to make GCD of 15, 96; and make sure that your program can calculate the GCD value.