Postfix Calculator

Tom Kelliher, CS 220

Due Nov. 11, 2011, with a milestone due Nov. 4

Simply, hand-compile the `postfix.c` program on the class home page to MIPS assembly and run under SPIM. Use `typescript` to record a session in which you run the expressions on the "Postfix Calculator Test" document (refer to the class web site) on your calculator in SPIM. Hand-in the typescript session and your **commented** MIPS source code.

Ground rules:

1. Frames and a frame stack must be used for all functions, including `main()`. This is crucial for `printVal()`, as this function is recursive. You may NOT implement `printVal()` non-recursively.

   Note that registers `$fp` and `$sp` are for the frame stack. Use `$t8` (see below) for the operand stack.

   The program makes use of an operand stack. This operand stack and the frame stack are **completely** independent of each other.

2. Implement the following `postfix.c` functions:

   ```
   int isEmpty(void);
   int isFull(void);
   void push(int item);
   int pop(void);
   int getValue(void);
   void compute(void);
   void output(void);
   void die(const char *s);
   void printVal(int v);
   int main();
   ```

3. Use mnemonic register names.

4. Adhere to the following MIPS conventions: call, register use, and memory use.

5. `$t8` and `$t9` may be used to permanently store the two external pointers (`stackPointer` and `next`).

Milestone **due** Nov. 4: Lay-out the frame structure and write the bodies of the functions, save for `main`. Do not consider the frame push/pop code for now. Hand in hard copy of frame maps and function bodies.