Chase Pinkerton (cpinke01)
Jake Seidman (jseid01)

asmcoding

Authors

- Chase Pinkerton and Jake Seidman
- (cpinke01) and (jseidm01)

Collaborative Work

- Thank you to the TAs!

Correctly Implemented

- Everything has been correctly implemented to our knowledge

Departures from the recommended calling convention

We mostly stayed with the recommended calling convention with these changes:

- Register 4 contains the value stack pointer, this was recommended as a nonvolatile general-purpose register. Most of the times this register was not edited, other than within

operations where values were popped off and an operation was executed and the result was pushed on the stack.
- Register 1 was sometimes used as a temporary register within certain labels. This was mostly used within operations where the return address was not needed since the program would always go to waiting after every operation. r1 was used for extra registers on most comparisons.

* These changes were made for operations.ums and calc40.ums, not print_d.ums

Print Module

- Our print module outputs the top value on the call stack by looping through it and using modulus of 10 and division by 10 to get each digit of the value from left to right until the end where we then loop through and output each value in the correct order

* We used r4 as a non-volatile register so our value stack is not changed

Value Stack

- Our value stack is stored in r4 and values are pushed onto the stack and combined if needed to form multiple digit numerals, these values are then popped off and pushed after completing whatever operation is given.

Sections

We have an init section for each of our .ums files. We use init in each file set r6 and r7 to temporary registers, and we also set r0 to zero ech time.

urt0.ums:

We have a data section that creates our call and value stacks and then allocates space for them using .space. In our init section here we set r2 to the end of the call stack, and r4 to the end of the valuestack. We also set r0 to 0 which will never be changed again.

calc40.ums:

We have a Read-only dats section in which we allocate space for the jump table. In our init section, we initalize the jump table so that all values jump to the label "input_error"

printd.ums:

We have the usual init section here along with a text section that contains the umasm code to print out a value.

operations.ums:

Our init section here just sets the temps and zeros r0. We then have a large text section that contains labels for each operation.

callmain.ums:

Contains an init section that calls main linking r1, then halts

Time Spent

– We spent 2 hours analyzing the assignment
– We spent 6 hours writing assembly code
– We spent 2 hours debugging the calculator