Homework 9

Due Monday by 11:59pm **Points** 20 **Submitting** on paper

Name:

Print this page, write your name, follow the instruction below, and return the completed page to me by 4/19/2019.

Part 1 (10 points)

Most x86-64 (and Y86-64) instructions transform the program state in a straightforward manner. Some unusual instruction combinations, however, require special attention. In Section 3.4.2 of the textbook, the x86-64 *pushq* instruction was described as decrementing the stack pointer and then storing the register at the stack pointer location. So, if we had an instruction of the form *pushq REG*, for some register *REG*, it would be equivalent to the code sequence

```
subq $8, %rsp
movq REG, (%rsp)
```

A. In light of analysis done in Practice Problem 4.7, does this code sequence correctly describe the behavior of the instruction *pushq %rsp*? YES NO Explain.

The operation pushq REG is given by R[%rsp] <-- R[%rsp] 8; M[R[%rsp]] <-- REG, which is exa

B. How could you rewrite the code sequence so that it correctly describes both the cases where *REG* is *%rsp* as well as any other register?

movq REG, -8(%rsp)subq

Part 2 (10 points)

Add to the table appearing in Figure 4.24 of the textbook by filling the column for the Y86-64 *pushq rA* instruction. The table is reproduced below for your convenience.

Stage	Computation	pushq rA
Fetch	icode, ifun rA, rB valC valP	icode:fun <-- M_1[PC]rA:rB
Decode	valA, srcA valB, srcB	valA <-- R[rA]valB <--
Execute	valE Condition codes	valE <-- valB + (-8)
Memory	Read/write	M_8[valE] <-- valA
Write back	E port, dstE M port, dstM	R[%rsp] <-- valE
PC update	PC	PC <-- valP

Part 3 (extra credit, 10 points)

Assume that we changed the semantic of the *pushq* instruction (and the *popq*, *call*, *ret* instructions to match the new definition) as follows

```
movq REG, (%rsp)
subq $8, %rsp
```

Does this change impact a pipelined implementation of the CPU in a significant way? YES NO Explain.