**CPSC 313**

Klaus Marte — 2141 5054

Nafis Abrar — 5806 8131

1) (pseudo code)

```
iopq V, rB
Fetch:
    iCd:iFn <- M1[PC]
    rA:rB <- M1[PC+1]
    valC <- M8[PC+2]
    valP <- PC+10
Decode:
    valA <- null
    valB <- R[rB]
    dstE <- rB
Execute:
    valE <- valB OP valC
    set CC
Memory:

Write Back:
    R[dstE] <- valE
PC update:
    PC <- valP
```

**CPSC 313**
Klaus Marte 2141 5054
Nafis Abrar 5806 8131

**call (rA)**

```
Fetch:
     iCd:iFn <- M1[PC]
     rA:rB <- M1[PC+1]
     valP <- PC+2
Decode:
     valA <- R[rA]
     valB <- R[%rsp]
     dstE <- %rsp
Execute:
     valE <- valB-8
Memory:
     M8[valE] <- valP
Write Back:
     R[dstE] <- valE
PC update:
     PC <- valA
```

**CPSC 313**

Klaus Marte 2141 5054

Nafis Abrar 5806 8131

**rmmovq rA, D(rB,8)**

```
Fetch:

     iCd:iFn <- M1[PC]

     rA:rB <- M1[PC+1]

     valC <- M8[PC+2]

     valP <- PC+10

Decode:
      srcA <- rA
      srcB <- rB

     valA <- R[srcA]

     valB <- R[srcB]

Execute:

     valE <- valB*8 + valC

Memory:

     M8[valE] <- valA

Write Back:


PC update:

     PC <- valP
```

**CPSC 313**

Klaus Marte
2141 5054

Nafis Abrar
5806 8131

**mrmovq D(rB,8), rA**

```
Fetch:

     iCd:iFn <- M1[PC]

     rA:rB <- M1[PC+1]

     valC <- M8[PC+2]

     valP <- PC+10

Decode:
     srcA <-rA
     srcB <-rB

     valA <- R[rA]

     valB <- R[rB]

     dstM <- rA

Execute:

     valE <- valB*8 + valC

Memory:

     valM <- M8[valE]

Write Back:

     R[dstM] <- valM

PC update:

     PC <- valP
```

**CPSC 313**

Klaus Marte Nafis Abrar

2141 5054 5806 8131

2) (implementations)
   See CPU.java.
3) (tests)
   All tests are processed as expected. The *call*, *mrmovq* and *rmmovq* instructions have been tested using their conventional syntax (to ensure that the required changes do not interfere) and the new syntax. The expected results are provided in the test files as comments.
   a) The test for iopq, we use 7 operations that is given. We make sure the arithmetic operations function properly.
      Test results for iopq.s:
      iaddq:value of %rax=9
      isubq:value of %rbx=2
      imulq:value of %rcx=18
      idivq:value of %rdx=2
      imodq:value of %r8=2
      iandq:value of %r9=2
      xorq: value of %r10=12(0xC)
   b) *call* calls an instruction at an address stored in a register. This called instruction moves a constant value into a given register and returns to the instruction the follows the *call* instruction. To show that the new implementation of *call* does not interfere with returns, another constant value is saved in a given register.
   c) *mrmovq* retrieves a value from a memory address to save to a given register. In our case, %rsi is expected to hold the value 0xe.
   d) *rmmovq* is similar to the test for the modified *mrmovq* instruction except that we are now expecting a specific value in a specific memory location.
4) (time required)
   8 hours