

## CS 308 — Homework #4

**Due Wednesday, April 21, in class**

1. From the textbook — page 191, no. 4.2.
2. From the textbook — page 302, no. 6.8 — omitting the case of 1's complement.
3. The “SEAS 16-bit floating point standard” is just like the “IEEE floating point standard” except that it uses 1 bit for the sign, 5 bits for the exponent, and 10 bits for the fraction (or mantissa). The exponent is in excess 15 representation.

a) What is the decimal representation of the following SEAS floating point number?

0 01001 0101000000

(b) Now go the other way. What is the “SEAS standard 16-bit” representation of the following decimal number? (show your answer in binary, all 16 bits of it)

- 50.75

---

4. Your company is designing the new MIPS (MIdget Processor with Stack) machine. The MIPS is an 8-bit zero-address machine, and has been formally specified by the abstract RTN below. Unlike the SRC machine, which contains all operands in a single instruction word, the MIPS stores operands adjacent to their instructions in memory. For all instructions that require an operand (i.e. push, pop, and branches), the operand is located in memory at the next byte after the instruction.

$PC\langle 7..0 \rangle : IR\langle 7..0 \rangle : SP\langle 7..0 \rangle : Z :$

$M[0..2^8 \dots -1]\langle 7..0 \rangle :$

(PC  $\leftarrow$  0 ; start)

start := fetch :

fetch := (IR  $\leftarrow$  M[PC] ; PC  $\leftarrow$  PC + 1 ; execute)

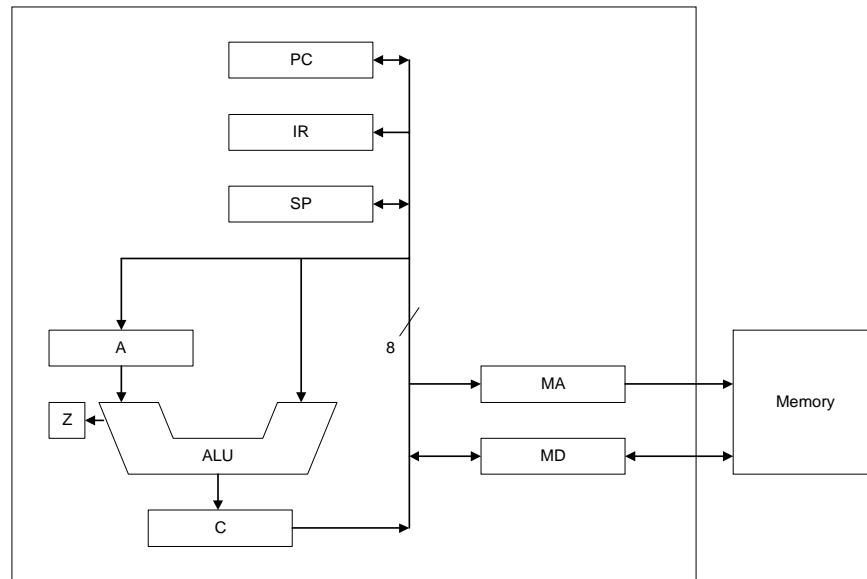
execute := (instruction ; fetch)

```

instruction := (
  push  (:= IR=0) → (SP ← SP + 1 ; M[SP] ← M[M[PC]] : PC ← PC + 1) :
  pushi (:= IR=1) → (SP ← SP + 1 ; M[SP] ← M[PC] : PC ← PC + 1) :
  pop   (:= IR=2) → (M[M[PC]] ← M[SP] : SP ← SP - 1 : PC ← PC + 1) :
  add   (:= IR=3) → (M[SP-1] ← M[SP-1] + M[SP] ; SP ← SP - 1 ; setcond) :
  sub   (:= IR=4) → (M[SP-1] ← M[SP-1] - M[SP] ; SP ← SP - 1 ; setcond) :
  br    (:= IR=5) → (PC ← M[PC]) :
  brz   (:= IR=6) → ( (Z=1) → PC ← M[PC] : (Z=0) → PC ← PC + 1) :
  brnz  (:= IR=7) → ( (Z=0) → PC ← M[PC] : (Z=1) → PC ← PC + 1)
)
setcond := ( (M[SP] = 0) → Z ← 1 : (M[SP] ≠ 0) → Z ← 0 )

```

(b) A possible one-bus data path design for the MIPS is depicted below:



The control signals for this data path are:

$PC_{out}$  - gate PC onto the bus

$PC_{in}$  - read into PC from bus

$IR_{in}$  - read into IR from bus

$SP_{out}$  - gate SP onto the bus

$SP_{in}$  - read into SP from bus

$A_{in}$  - read into A from bus

$C_{out}$  - gate C onto the bus

$C_{in}$  - read into C from bus

$MA_{in}$  - read into MA from bus

Read - read from memory

$MD_{out}$  - gate MD onto the bus

$MD_{in}$  - read into MD from bus

$MD_{wr}$  - send MD to memory

$MD_{rd}$  - read memory value into MD

INC - ALU increments bus input

DEC - ALU decrements bus input

ADD - ALU adds inputs, sets Z

SUB - ALU subtracts A from bus input, sets Z

C=B - ALU passes through bus input unchanged

Write - write to memory

Using the above data path design and the abstract RTN specification of the MIPS ISA given in part (a), write the concrete RTN and control sequences for the following; (make a table with the headings shown below for each part)

Part A: The fetch cycle for the MIPS machine:

Time Step	Concrete RTN	Control sequence

Part B: The pop instruction (don't repeat the fetch cycle above):

Time Step	Concrete RTN	Control sequence

Part C: The add instruction (omit the fetch cycle):

Time Step	Concrete RTN	Control sequence

Part D: The brz instruction (omit the fetch cycle):

Time Step	Concrete RTN	Control sequence

Pledge that you did this homework on your own.