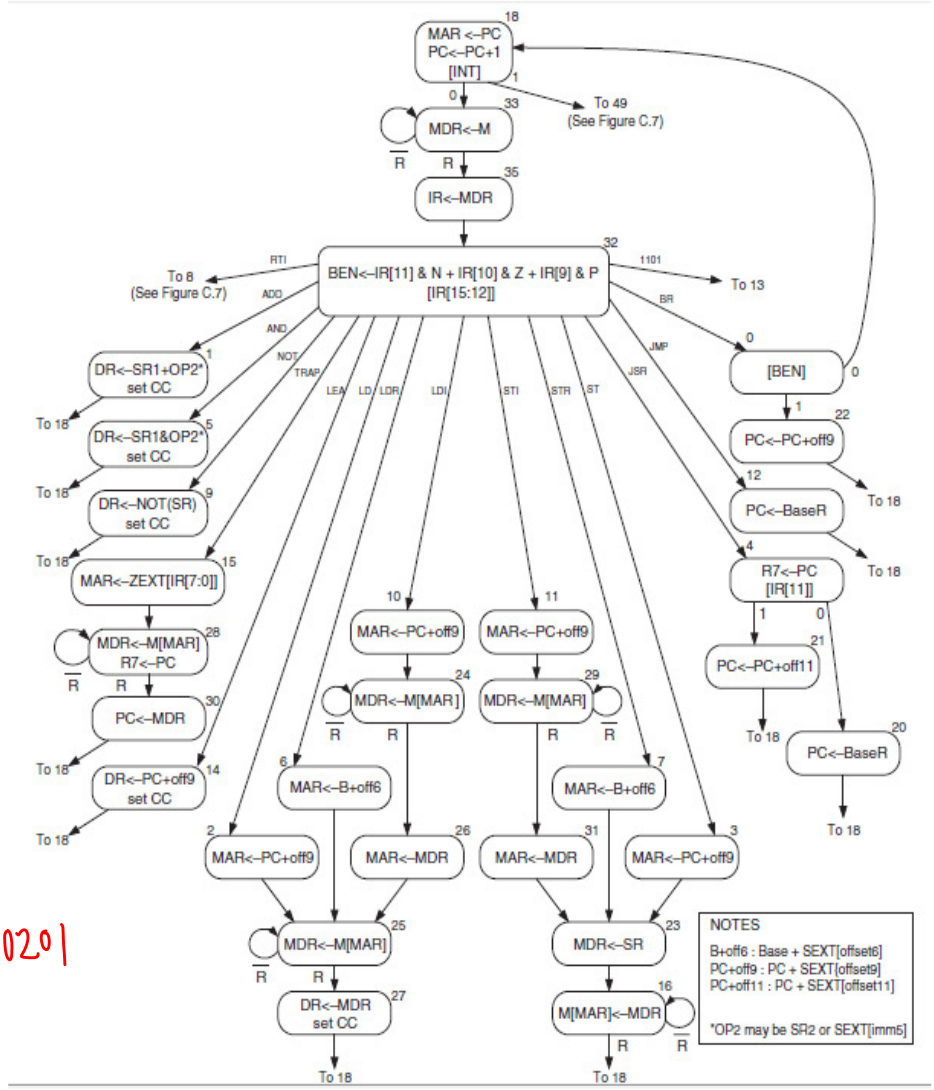
final exam

COSC-120, Computer Hardware Fundamentals, fall 2012
Computer Science Department
Georgetown University

NAME _____

Open books, open notes (laptops included). **Answers w/o explanation get 0 credit**. Credit is based on **your explanation, not correctness**. Show and explain **all** your work from first thoughts to final answer.

I.
At right is part of the LC3 controller's state-transition diagram. The parts missing handle interrupts (state 49), and exceptions (states 8 and 13). The questions that follow assume the controller has just entered state 18. Shown below are the relevant portions of the LC3's state. Register values and addresses are shown in hex, except for the PSR, which is shown as bits.



0200 PC

0 - - - - - 1 1 1 - - - - - 0 1 0 PSR

Memory at address 0200

200	LD R1, 9
201	AND R2, R2, 0
202	ADD R2, R2, R1
203	BRz 2
204	JSR 300
205	BRnzp 1
206	JSR 500
207	TRAP 25
208	0000
209	0000
20A	0201
20B	D000
20C	0FFF

R1 ← 0201

Memory content is translated into LC3 assembly language, except that offsets are shown as hex values instead of labels. Data words are also shown as hex values.

Q. Which instructions change the PSR content, according to the state-transition diagram?

In the diagram, "set CC" indicates a state that changes the CC in the PSR (states 1, 5, 9, 14, 27). the PSR content also includes Privilege (PSR[15]) and Priority (PSR[10:8]), but states that change these are not shown here.

Q. A clock cycle is 1 ns. Accessing memory takes 10 clock cycles; e.g., the controller remains in state 33 transitioning on the "R=0" arc 10 times before transitioning on the "R=1" arc to state 35. How long does it take to complete an instruction fetch? That is, given we have just transitioned to state 18, how long before we transition to state 32?

1 ns / state change + 10 ns in state 33 ⇒ 13 ns elapse until we enter state 32.

Q. How many clock cycles before the PC first contains a value greater than 02FF?

this address is not local: it must be a non-local jump. That is either JSR 300 or JSR 500, or TRAP. JSR 300 : PC ← 205 + 300. The BR_3 is not taken because ADD R2,R2,R1 is not zero: LD (13+14) ns, AND (13+2) ns, ADD (13+2) ns, BR_3 (13+2) ns, JSR (13+3) ns

$= 5(13) + (14+2+2+2+3) = 65 + 23 = 81$ cycles

Q. How many words of memory are represented above by the program and its data? This includes every word of the memory content shown. How many bits, in total?

Every instruction is one word → 8 words for instructions. Data is 5 words → 13 words $\left(\frac{16\text{ bit}}{1\text{ wrd}}\right)$ = 13×16 bits

Q. Assume memory address 0025 contains 0200. Will the program above ever make a jump outside of the range 0200 to 1000? That is, is there any instruction shown above that will change the PC to have a value less than 0200 or greater than 1000? If so, which instruction?

The BR_3 is local, JSRs jump to ≈ (200 + 300) or (200+500) < 800. Trap jumps to 0200 because vector at 0025 is 0200. So no jumps are further than +1000, and no jumps go below 200.

Q. Trace out the program's execution instruction-by-instruction: Just after completing an instruction's execution (just after the LC3 next enters state 18), show (1) the CC bits, (2, 3) the number of states entered and the number of ns elapsed since last entering state 18. Stop tracing when the PC first contains 0208.

instr, executed	N Z P	# states	# ns
LD	0 0 1	7	7+10+10
AND	0 1 0	5	5+10
ADD	0 0 1	5	5+10
BR_3	0 0 1	5	5+10

not sure these set executed because not sure JSR 300 returns. Also don't know NZP after JSR.

instr, executed	N Z P	# states	# ns
JSR 300	0 0 1	5	5+10
BRnzp	???	6	6+10
TRAP	???	1	1

PC ← PC+1 in state 18 ⇒ = 208

Q. Suppose again that the memory location at address 0025 contains 0200. Also assume the instructions at addresses 0200+**5**+300 and 0200+**7**+500 are JMP R7. Will this program enter an infinite loop?

oops

oops

Both JSR jmps return, BRnzp is taken, so TRAP executes. This causes jump to 0200, and we are starting all over again.

⇒ This is an ∞-loop.

Q. What is the content of R2 just after execution of the ADD instruction? Given the assumptions of the above question, will execution ever reach **0200+7+**500?

oops

R2 ← R2 + R1 (0) (0201) So R2 = 0201. The BRz will not be taken,

JSR 300 will execute and BRnzp +1 will jump over JSR 500. So PC never has 0700 (but wouldn't anyway because jump is to 0205+500 = 0705)

Q. Suppose Memory[0025] = 0211 **20c**. Will the program's execution ever cause a transition to state 13? Recall the decimal=hex=binary equivalents:
10=A=1010, 11=B=1011, 12=C=1100, 13=D=1101, 14=E=1110, and 15=F=1111.

oops

The TRAP will jump to 0211, which is not local to the above code. I have no idea what is there, so can't say what will happen. But nothing in this code will cause an illegal opcode exception (opcode = 1101 = D). But, assuming 211 is 20c, Then D000 is fetched, giving opcode exception to state 13.

Q. If instead Memory[0025] = 0208, will the program's execution ever cause a transition to state 13?

A jump to 208 executes a BR that is never taken because x0000 has opcode 0000 and nzp = 000. This is a NOP. Same at 0209. At 20A is a BR$_{0000}$ P$_{0010}$ +1$_{0000\ 0001}$. this BRp +1 might be taken, jumping over xD000 at 20B. But it might not, and an illegal opcode exception would occur.

Q. In all the questions above, any possibility of a transition to state 49 is ignored; that is, we assumed that interrupts would not occur. What information given above about the state of the machine guarantees that interrupts cannot occur?

The PSR.Priority = 7. No interrupt can occur because the code is executing at highest priority already.

II. Suppose we have a physical 32-bit LC3 machine (32-bit data path, 32-bit registers, 32-bit addresses, 32-bit memory words). Suppose its memory is byte-addressable (a 32-bit word can be fetched starting from any byte location), and it is little-endian (the lowest addressed byte is the least significant 8 bits). Suppose the memory content is,

address	content byte (in hex)
12345678	30
12345679	31
1234567A	32
1234567B	33
1234567C	34
1234567D	35
1234567E	36
1234567F	37
12345680	38

ASCII code	char
x30	'0'
x31	'1'
...	...
x39	'9'

Because it is little-endian, if R6 = 12345678, executing these instructions,

```
   LDR R1, R6, 0
   LDR R2, R6, 1
```

results in,

R1 = 33323130
R2 = 34333231

Q. Suppose we order our LC3 with the maximum amount of memory installed. What is the physical memory size in 32-bit words? In bytes?

32-bit memory addresses → 2^{32} locations, each referencing one Byte.
⇒ $2^2 \cdot 2^{30} = 4GB$. $4GB\left(\frac{Word}{4B}\right) \Rightarrow 1G Word$.

Q. Recall that when we send a word to the LC3's DDR (Display Data Register), only the low byte is displayed, the upper bytes are ignored. I/O is mapped to the 2^9 (512) addresses at the big-address end of memory, similarly to the original LC3 with addresses extended to 32 bits. Consider this code,

```
        AND R3, R3, 0
        ADD R3, R3, 8     ;--- initialize loop counter
LOOP:   LD  R1, R6, 0
        STI R1, DDR
        ADD R1, R1, 1
        ADD R3, R3, -1
        BRp LOOP
        TRAP x25
DDR:    .FILL xFFFFFE06   ;--- Pointer to DDR
```

R3 ← 0
R3 ← 8
R1 ← M[12345678] = 33323130
← x30 gets printed ⇒ '0' printed
R1 ← 12345679
R3--

Loops 8 times.
R1 advances 1 digit each loop: '01234567' is printed.

But, R1 always reloaded at LOOP ⇒ 00000000

Show the display's output, assuming the output cursor was originally in the upper-left character position.

Display

8 x (ascii x30) → 8 x '0'

Q. What would be in R3 after execution of this program fragment?

```
LDR R3, R6, 3
ADD R3, R3, R3
ADD R3, R3, R3
ADD R3, R3, R3
ADD R3, R3, R3
```

R3 ← 36 35 34 33
L shift
L shift
L shift
L shift
R3 ← 63534330

Q. What does the following code display?

```
        LDR  R0, R6, 0      ;-- R0 <=== Mem[ R6=12345678 ]
        LD   R4, MASK       ;-- R4 <=== MASK
        LD   R5, CNTR       ;-- R5 <=== CNTR
        AND R1, R1, 0       ;-- R1 <=== 0
LOOP:   AND R3, R4, R0      ;-- is current bit 0?
        BRz SHIFT
        ADD R1, R1, 1
SHIFT:  ADD R1, R1, R1
        ADD R4, R4, R4
        ADD R5, R5, R5
        BRnp LOOP           ;-- if not done, do next bit
DONE:   STI R1, DDR
        TRAP x25            ;-- HALT

MASK:   .FILL x00000100     ;--- Note ":" is ignored for labels.
CNTR:   .FILL x01000000
DDR:    .FILL xFFFFFE06
```

R0 = 33323130
R4 → 0000 0000 0000 0000 0000 0001 0000 0000
R1 →
0011 0011 0011 0010 0011 0010 0011 0000
0100 1100 (4 A)
mask for bit 8
if 1, add 1 to R1
R1 ++
L shift R1
reverses bits
L shift mask
L shift CNTR
stops when R5's 1 L shifted out → 8 shifts
print low 8 bits of R1
mask bit 8
R1 gets 8 bits of R0[15:8]
→ prints '1' (?) but reversed
→ x4A (some char?)

Q. Treating VALUE as an integer variable, what is the effect of this code?

```
AND R0, R0, 0
ADD R0, R0, 1
ADD R0, R0, R0
ADD R0, R0, R0
NOT R0, R0
LD   R1, VALUE
ADD R1, R1, R0
ST   R1, VALUE

VALUE:  .FILL x00001234
```

R0 ← 0
R0 ← 1
L shift R0
L shift R0 = x0004
FFFB
0100
1011
as 2's complement, this is -5
1234
122F = R1 ← VALUE - 5

Don't count?

Q. We want to turn on the n-th bit of R0, where n is a positive integer in R5, without changing other bits of R5. What are the max and min n can be, and still make sense (show in decimal and hex)? Complete the code at right. Assume R1 contains 1.

```
L1:  BRz L2
     ADD R1, R1, R1
     ADD R5, R5, -1
     BRnzp L1
L2:  NOT R0, R0
     NOT R1, R1
     AND R0, R0, R1
     NOT  R0, R0
```

oops, missing code!
Bits of R0 are $b_{31} b_{30} \ldots b_1 b_0$
max $n = x31 = [(3 \cdot 16) + 1]_{10} = 49_{10}$
min $n = x0 = 0_{10}$
LShifts R1 to bit position R5
This De Morgan dual of OR
Turns on bit.
oops (R0)

Q. In order to get a result **value** of 1, what positive **value** would we add to the **value** represented by the content of R1? Assume R1's content is a number encoded in 32-bit floating point format.

bits in R1: 0 01111111 11100000000000000000000

Recall that the 32-bit IEEE floating-point standard has 1 sign bit, 8 exponent bits in excess-127 code, and the remaining bits as the fractional part.

Q. Encode the value 1/16 in IEEE 32-bit FP format.

Q. The bits of R1 shown above are 3FF00000 in hex. Treated as a 2's complement number, what **value** should we add to R1 to get the result **value** of 2^30 (1 G)?

III. All following questions assume the original, 16-bit, 16-bit-word addressable, LC3.

Q. Fill in the values in the assembler's symbol table (at right) after the first pass of assembly of this code:

```
        .ORIG x3000
here
        LD R1, there
        TRAP x25
there
        .FILL x1234
        .END
```

SYMBOL TABLE

STRING	VALUE
"here"	3000
"there"	3002

Q. Suppose OS code at x0200 sets up the Vector Tables, sets R6=x0000, enables keyboard interrupts, and jumps to x3000 setting PSR[15] = 1 (user mode) and PSR[10:8] = 000 (priority). What problem might occur? User's code is:

```
L: ADD R6, R6, -1
   BRnzp L
```