Name: SOLUTIONS

CPE 233 Winter 2019 Midterm

1. RAT ASSEMBLY MEMORY CONTENTS – 10 pts

Assume the register file and scratch pad memory have the initial values below. Recall that arithmetic instructions load the Carry and Zero flags and Logic instructions clear the Carry flag and load the Zero flag. All other instructions do not affect the flags.

ROL: Rd ← Rd(6:0) & Rd(7), C ← Rd(7)
LSR: Rd ← C & Rd(7:1), C ← Rd(0)
ASR: Rd ← Rd(7) & Rd(7) & Rd(6:1), C ← Rd(0)

```
.CSEG
.ORG          0x30

   30 LD      R1,  (R4)
   31 ADDC    R2,  R4
   32 EXOR    R30, skip ← 0x36
   33 ST      R3,  (0xFD)
   34 BREQ    skip
   35 ROL     R5
36 skip: LSR  R0
```

Modify the register file, scratch ram, cflag and zflag according to the execution of the program. Write your answers in hex.

0	~~0x04~~ 0x02
1	~~0x62~~ 0xF9
2	~~0xFF~~ 0x01
3	0x00
4	0x02
5	~~0x11~~ 0x22
.	
.	
30	~~0x32~~ 0x04
31	0x98

Register File

0	0x23
1	0x11
2	0xF9
3	0x1F
4	0x54
.	
.	
253	~~0x0B~~ 0x00
254	0x28
255	0x54

Scratch Pad Memory

~~0~~ ~~1~~ 0

CFlag

0

ZFlag

2. CALCULATE THE TIMING OF THE FOLLOWING RAT PROGRAM

Recall the RAT CPU runs at 50MHz, and each instruction takes two clock cycles, so each instruction takes 40ns. Write an equation (in terms of A and B) that calculates the amount of time it takes to execute the following program. You can assume A and B are not zero.

8 pts

```
.CSEG
.ORG 0x01

main:      ✓MOV   R2, A

Out1:       ADD   R20, 0x01
            SUB   R2, 0x01
            MOV   R3, B

Out2:       ADD   R21, 0x01
            SUB   R3, 0x01         B        A
            BRNE  Out2

            OR    R2, 0x00
            BRNE  Out1
           ✓BRN   End

            SUB   R21, 0x03
            SUB   R22, 0x03    X skipped

End:       ✓OUT   R21, 0xFF
           ✓OUT   R22, 0xFF
```

$$\text{time} = \left[(3B + 5)A + 4\right] 40\text{ns}$$

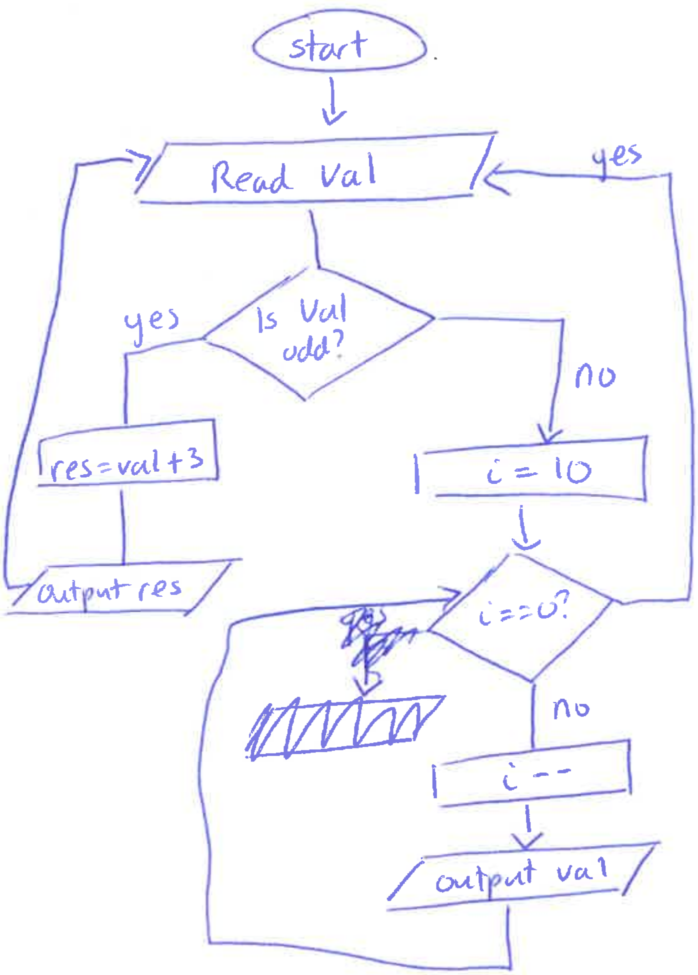
3. DRAW A FLOWCHART AND WRITE RAT CODE

- 10 pts

Draw a flowchart and write a short RAT assembly program to implement the following:

Read in a value from Port 0x22. If the value is odd, add 3 to the value and output the result to Port 0x23. Otherwise, output the value to port 0x24 ten times (must use a loop). Have your program repeat indefinitely. Use R0 for the input value and R1 for the output value.

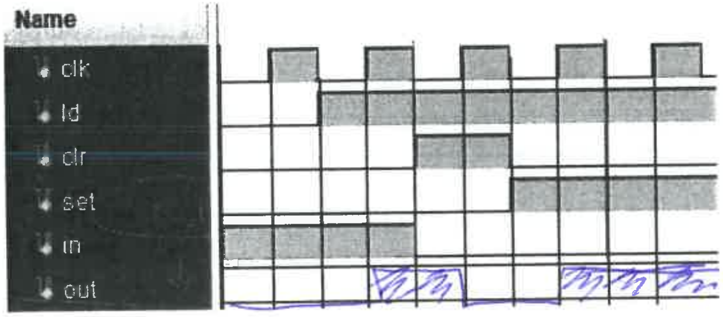
```
.EQU      IN_PORT = 0x22
.EQU      OUT_PORT = 0x24
.CSEG
.ORG 0x01

Start:
          IN    R0, IN-PORT
          MOV   R1, R0
          TEST  R0, 0x01
          BREQ  even
          ADD   R1, 3
          OUT   R1, 0x23
          BRN   start

even:     MOV   R2, 10
loop:     CMP   R2, 0
          BREQ  Start
          OUT   R1, OUT_PORT
          ~~BRN   loop~~
          SUB   R2, 1
          BRN   loop
```



4. RAT TIMING DIAGRAMS

4 pts

a. Fill in all of the boxes for output (out) of this rising-edge triggered flag register. Of the control signals, set has the highest precedence and ld has the lowest precedence.



b. Fill in the 18-bit machine code and complete the timing diagram of the RAT CPU for the following code. Fill in any box that is non-zero.

```
.CSEG
.ORG 0x23
main:   IN      R2, 0x64
        ADD     R2, 0xFF
        SUBC    R2, 0x32
        BRN     main
```

4 pts

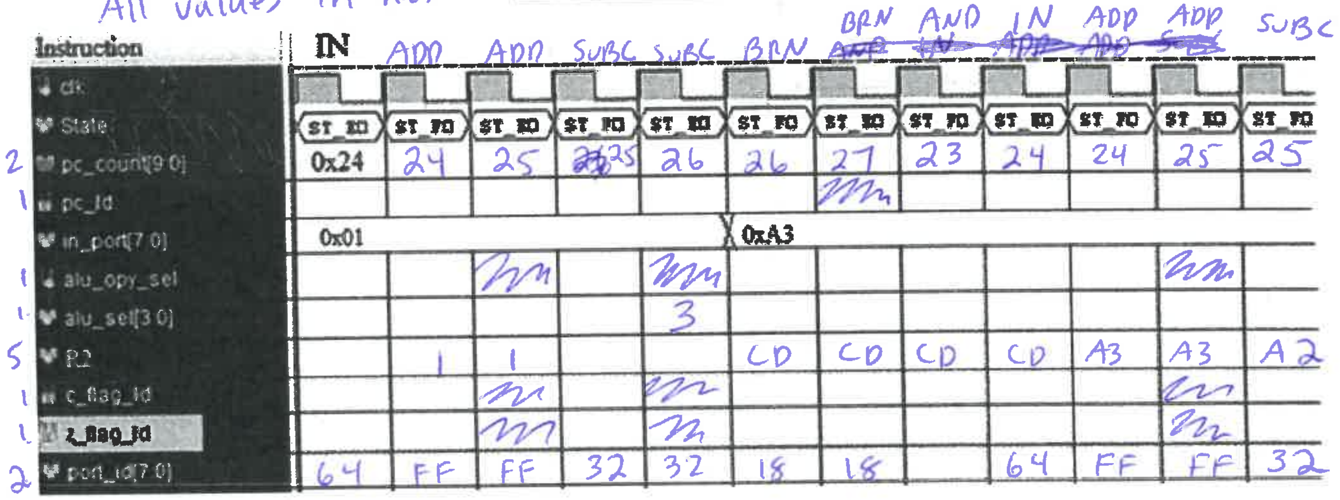
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	0	0	1	0	0	1	1	0	0	1	0	0
1	0	1	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1
1	0	1	1	1	0	0	0	1	0	0	0	1	1	0	0	1	0
0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0

All values in hex

~~15~~ 14 pts

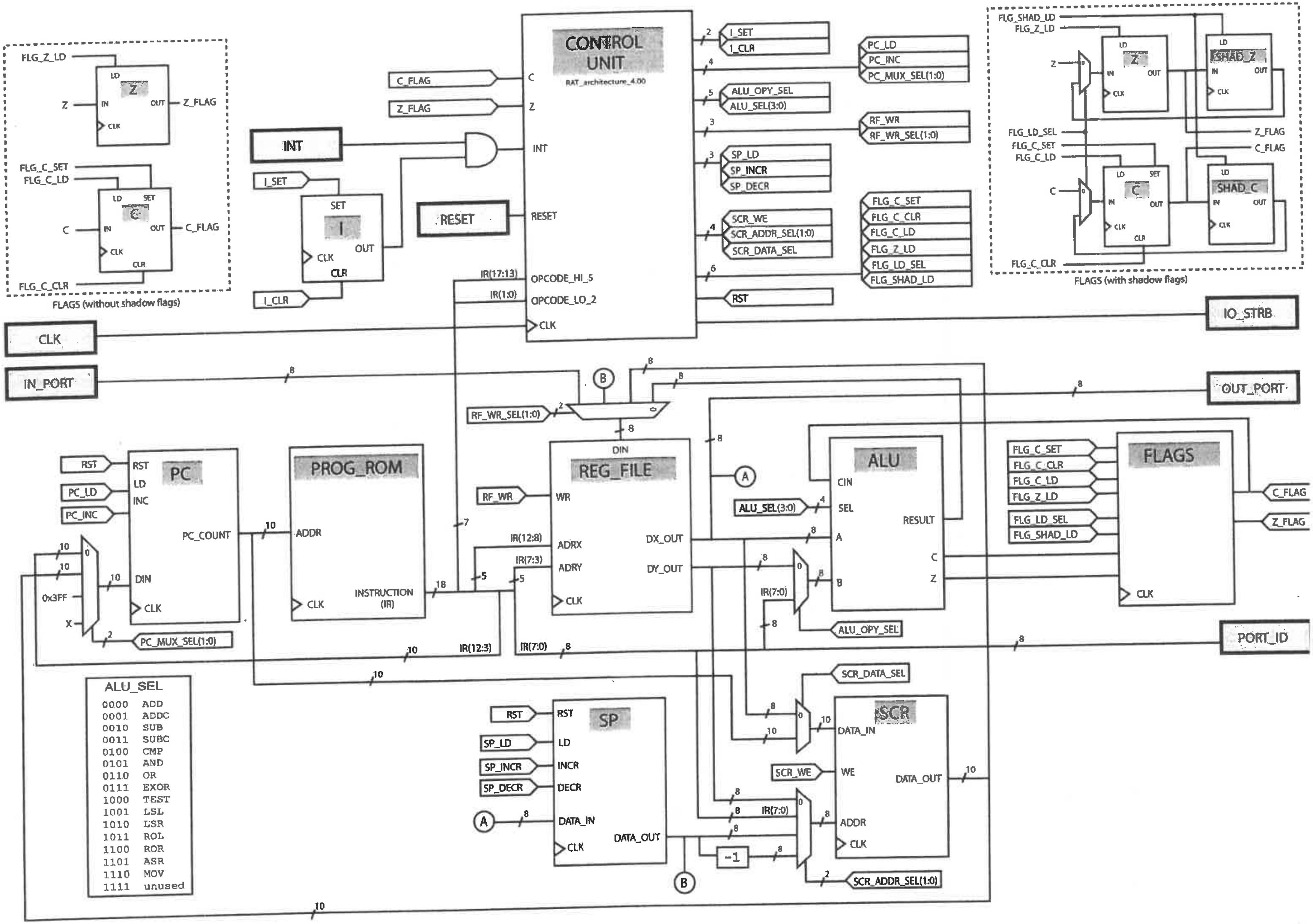
(then half)

7pts



	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AND AND rx, imm	1	0	0	0	0	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
OR OR rx, imm	1	0	0	0	1	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
EXOR EXOR rx, imm	1	0	0	1	0	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
TEST TEST rx, imm	1	0	0	1	1	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
ADD ADD rx, imm	1	0	1	0	0	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
ADDC ADDC rx, imm	1	0	1	0	1	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
SUB SUB rx, imm	1	0	1	1	0	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
SUBC SUBC rx, imm	1	0	1	1	1	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
CMP CMP rx, imm	1	1	0	0	0	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
IN IN rx, imm	1	1	0	0	1	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
OUT OUT rx, imm	1	1	0	1	0	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
MOV MOV rx, imm	1	1	0	1	1	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
LD LD rx, imm	1	1	1	0	0	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k
ST ST rx, imm	1	1	1	0	1	rX	rX	rX	rX	rX	k	k	k	k	k	k	k	k

	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRN BRN label	0	0	1	0	0	aa	aa	aa	aa	aa	aa	aa	aa	aa	aa	-	0	0
CALL CALL label	0	0	1	0	0	aa	aa	aa	aa	aa	aa	aa	aa	aa	aa	-	0	1
BREQ BREQ label	0	0	1	0	0	aa	aa	aa	aa	aa	aa	aa	aa	aa	aa	-	1	0
BRNE BRNE label	0	0	1	0	0	aa	aa	aa	aa	aa	aa	aa	aa	aa	aa	-	1	1
BRCS BRCS label	0	0	1	0	1	aa	aa	aa	aa	aa	aa	aa	aa	aa	aa	-	0	0
BRCC BRCC label	0	0	1	0	1	aa	aa	aa	aa	aa	aa	aa	aa	aa	aa	-	0	1



ALU_SEL	
0000	ADD
0001	ADDC
0010	SUB
0011	SUBC
0100	CMP
0101	AND
0110	OR
0111	EXOR
1000	TEST
1001	LSL
1010	LSR
1011	ROL
1100	ROR
1101	ASR
1110	MOV
1111	unused