

XueJin Cui (S5876095)

INSTRUCTIONS

1. Submission is only via Themis for the practical exercises and via Brightspace for the theoretical exercises. Deadlines are strict.
 2. The exercises in this assignment add up to 100 points. To calculate your grade simply divide the number of points by 10.
 3. You must submit a pdf typeset in (La)TeX (no handwritten solutions) using **this** template.
 4. Seeking solutions from the internet, from any external resource, or from any other person is prohibited.
 5. Please note that the course lecturer reserves the right to ask the student submitting the assignment to explain the answers to any or all questions. If the student is unable to provide a satisfactory answer then that question may receive partial/no credit.
 6. Of course, university policies on plagiarism always apply. In particular, any suspected plagiarism will be reported to the Board of Examiners.
-

1. You are given the following LC3 ASM program! For this exercise, you have two tasks: **(10 points)**

1. Write the symbol table (use an absolute address, not offsets)
2. Manually assemble (aka convert to the internal LC3 representation) the instructions at A, B and D. Both binary and hex are fine.

```
.ORIG x3000
    AND R0, R0, #0
A   LD R1, E
    AND R2, R1, #1
    BRp C
B   ADD R1, R1, #-1
C   ADD R0, R0, R1
    ADD R1, R1, #-2
D   BRp C
    ST R0, F
    TRAP x25
E   .BLKW 1
F   .BLKW 1
    .END
```

Solution:

1. **Symbol Table:**

Symbol	Absolute Address
A	x3001
B	x3004
C	x3005
D	x3007
E	x300A
F	x300B

2. Manual Assembly:

- **Instruction A (LD R1, E):**

- Opcode: 0010, DR: 001 (R1), PCOffset9: $x300A - x3002 = 8$ (000001000)
- Binary: 0010001000001000

- **Instruction B (ADD R1, R1, #-1):**

- Opcode: 0001, DR: 001 (R1), SR1: 001 (R1), Imm5: -1 (11111)
- Binary: 0001001001111111

- **Instruction D (BRp C):**

- Opcode: 0000, nzp: 001 (p), PCOffset9: $x3005 - x3008 = -3$ (111111101)
- Binary: 0000001111111101

2. For this exercise we will assume that two new operations have been added to LC3: `PUSH` and `POP`. The instruction `PUSH Rx` pushes the value in Register x onto the stack. `POP Rx` removes a value from the stack and loads it into Rx . You are also given a list of the instructions which were executed, however some of the registers went missing. Fill out what the correct registers would be! **(10 points)**

BEFORE		AFTER	
R0	x0000	R0	x1111
R1	x1111	R1	x1111
R2	x2222	R2	x3333
R3	x3333	R3	x3333
R4	x4444	R4	x4444
R5	x5555	R5	x5555
R6	x6666	R6	x6666
R7	x7777	R7	x4444

Operations:

```
PUSH R4
PUSH (a)
POP (b)
PUSH (c)
POP R2
POP (d)
```

Solution:

By analyzing the transition from the initial register values to the final states:

1. **PUSH R4:** The value $x4444$ is pushed onto the stack. [Stack: $x4444$]

2. **PUSH (a):** The AFTER table shows $R0$ becomes $x1111$. Since $x1111$ was the original value of $R1$, we must push $R1$ to eventually pop it into $R0$. Thus, **(a) = R1**. [Stack: $x4444, x1111$]
3. **POP (b):** The top of the stack ($x1111$) is removed. Since $R0$ is the register that changed to $x1111$, the value was popped into $R0$. Thus, **(b) = R0**. [Stack: $x4444$]
4. **PUSH (c):** The AFTER table shows $R2$ becomes $x3333$. Since $x3333$ was the original value of $R3$, we must push $R3$. Thus, **(c) = R3**. [Stack: $x4444, x3333$]
5. **POP R2:** The top of the stack ($x3333$) is popped into $R2$, which matches the AFTER state for $R2$. [Stack: $x4444$]
6. **POP (d):** The remaining value on the stack ($x4444$) is popped. The AFTER table shows $R7$ changed from $x7777$ to $x4444$, meaning the value was popped into $R7$. Thus, **(d) = R7**.