

Department of Electrical and Computer Engineering

The University of Texas at Austin

EE 306, Fall 2023

Problem Set 6

Due: Not to be turned in

Yale N. Patt, Instructor

TAs: Chester Cai, Kayvan Mansoorshahi, Sophia Jiang, Jaeyoung Park, Varun Arumugam, Anna Guo, Nadia Houston, Asher Nederveld, Edgar Turcotte

Note: This problem set is unusually long, and is not to be turned in. We have put it together and handed it out to give you some challenging examples to help you prepare for the final exam.

1. Jane Computer (Bob's adoring wife), not to be outdone by her husband, decided to rewrite the TRAP x22 handler at a different place in memory. Consider her implementation below. If a user writes a program that uses this TRAP handler to output an array of characters, how many times is the ADD instruction at the location with label A executed? Assume that the user only calls this "new" TRAP x22 once. Is it ok to call TRAP x21 within this "new" Trap routine? Explain why or why not in 20 words or fewer.

; TRAP handler
; Outputs ASCII characters stored in consecutive memory locations.
; R0 points to the first ASCII character before the new TRAP x22 is called.
; The null character (x00) provides a sentinel that terminates the output sequence.

```
                .ORIG x020F
START          LDR R1, R0, #0
                BRz DONE
                ST R0, SAVER0
                ADD R0, R1, #0
                TRAP x21
                LD R0, SAVER0
A              ADD R0, R0, #1
                BRnzp START
DONE           RTI
```

```
SAVER0    .BLKW #1
          .END
```

2. (Adapted from 9.16)

- How many TRAP service routines can be implemented in the LC-3? Why?
- Why must a RTI instruction be used to return from a TRAP routine? Why won't a BRnzp (unconditional BR) instruction work instead?
- How many accesses to memory are made during the processing of a TRAP instruction?

3. (Adapted from 8.15)

- What does the following LC-3 program do?

```

          .ORIG x3000
          LD R3 , A
          STI R3, KBSR
AGAIN     LD R0,B
          TRAP X21
          BRnzp AGAIN
A         .FILL X4000
B         .FILL X0032
KBSR      .FILL XFE00
          .END
```

- If someone strikes a key, the program will be interrupted and the keyboard interrupt service routine will be executed as shown below. What does the keyboard interrupt service routine do?

```

          .ORIG X1000
          LDI R0,KBDR
          TRAP X21
          TRAP X21
          RTI
KBDR      .FILL XFE02
          .END
```

- Finally, suppose the program of part a started executing, and someone sitting at the keyboard struck a key. What would you see on the screen?

4. (9.34) What does the following LC-3 program do?

```
                .ORIG x3000
                LD  R0, ASCII
                LD  R1, NEG
AGAIN           LDI R2, DSR
                BRzp AGAIN
                STI R0, DDR
                ADD R0, R0, #1
                ADD R2, R0, R1
                BRnp AGAIN
                HALT
ASCII           .FILL x0041
NEG             .FILL xFFB6
DSR             .FILL xFE04
DDR             .FILL xFE06
                .END
```

5. (Adapted from 10.1)

What are the defining characteristics of a stack? Give two implementations of a stack and describe their differences.

6. Consider the following LC-3 assembly language program. Assuming that the memory locations DATA get filled before the program executes, what is the relationship between the final values at DATA and the initial values at DATA?

```
                .ORIG  x3000
LEA             R0, DATA
                AND  R1, R1, #0
                ADD  R1, R1, #9
LOOP1           ADD  R2, R0, #0
                ADD  R3, R1, #0
LOOP2           JSR  SUB1
                ADD  R4, R4, #0
                BRzp LABEL
                JSR  SUB2
LABEL           ADD  R2, R2, #1
                ADD  R3, R3, #-1
```

```

        BRp    LOOP2
        ADD    R1, R1, #-1
        BRp    LOOP1
        HALT
DATA    .BLKW  #10
SUB1    LDR    R5, R2, #0
        NOT    R5, R5
        ADD    R5, R5, #1
        LDR    R6, R2, #1
        ADD    R4, R5, R6
        RET
SUB2    LDR    R4, R2, #0
        LDR    R5, R2, #1
        STR    R4, R2, #1
        STR    R5, R2, #0
        RET
        .END

```

7. During the initiation of the interrupt service routine, the N, Z, and P condition codes are saved on the stack. By means of a simple example show how incorrect results would be generated if the condition codes were not saved. Also, clearly describe the steps required for properly handling an interrupt.
8. The program below counts the number of zeros in a 16-bit word. Fill in the missing blanks below to make it work.

```

        .ORIG x3000
        AND    R0, R0, #0
        LD     R1, SIXTEEN
        LD     R2, WORD
A       BRn    B
        _____
B       _____
        BRz    C
        _____
C       BR     A      ; note: BR = BRnzp
        ST     R0, RESULT
        HALT

```

```

SIXTEEN    .FILL #16
WORD       .BLKW #1
RESULT     .BLKW #1

            .END

```

After you have the correct answer above, what one instruction can you change (without adding any instructions) that will make the program count the number of ones instead?

9. Fill in the missing blanks so that the subroutine below implements a stack multiply. That is it pops the top two elements off the stack, multiplies them, and pushes the result back on the stack. You can assume that the two numbers will be non-negative integers (greater than or equal to zero) and that their product will not produce an overflow. Also assume that the stack has been properly initialized, the PUSH and POP subroutines have been written for you and work just as described in class, and that the stack will not overflow or underflow.

Note: All blanks must be filled for the program to operate correctly.

```

MUL         _____
            ST R0, SAVER0
            ST R1, SAVER1
            ST R2, SAVER2
            ST R5, SAVER5
            AND R2, R2, #0
            JSR POP
            ADD R1, R0, #0
            JSR POP
            ADD R1, R1, #0
            _____

AGAIN       ADD R2, R2, R0
            _____
            BRp AGAIN
DONE        ADD R0, R2, #0
            JSR PUSH
            _____
            LD R0, SAVER0

```

```

LD R1, SAVER1
LD R2, SAVER2
LD R5, SAVER5
RET

```

10. The program below calculates the closest integer greater than or equal to the square root of the number stored in NUM, and prints it to the screen. That is, if the number stored in NUM is 25, "5" will be printed to the screen. If the number stored in NUM is 26, "6" will be printed to the screen. Fill in the blanks below to make the program work.

Note: Assume that the value stored at NUM will be between 0 and 81.

```

                .ORIG x3000
                AND R2, R2, #0
                LD R3, NUM
                BRz OUTPUT
                NOT R3, R3
                ADD R3, R3, #1
OUTLOOP        ADD R2, R2, #1

                _____
INLOOP         AND R1, R1, #0
                ADD R1, R1, R2
                ADD R0, R0, #-1
                BRp INLOOP

                _____
OUTPUT         BRn OUTLOOP
                LD R0, ZERO

                _____
                TRAP x21
                HALT
NUM            .BLKW 1
ZERO          .FILL x30
                .END

```

11. The figure below shows the part of the LC-3 data path that deals with memory and I/O. Note the signals labeled A through F. A is the memory enable signal, if it is 1 memory is enabled, if it is 0, memory is disabled. B, C, and D are the load enable signals for the Device Registers. If the load enable signal is 1, the register

PC	Instruction	Access	MAR	A	B	C	D	E[15:0]	F[1]	F[0]	MDR
x3000	LD R0, x9	1	x3000					x2009			
		2									
		3									
x3001	LDR R0, R0, #0	1									
		2									
		3									

12. Note: This problem is NOT easy. In fact, it took me a while to solve it, and I am supposed to be an expert on 306 material. So, if you are struggling to pass this course, I suggest you ignore it. On the other hand, if you are a hot shot and think no problem is beyond you, then by all means go for it. We put it on the problem set to keep some of the hot shots out of mischief. We would not put it on the final, because we think it is too difficult to put on the exam.

A programmer wrote this program to do something useful. He, however, forgot to comment his code, and now can't remember what the program is supposed to do. Your job is to save him the trouble and figure it out for him. In 20 words or fewer tell us what valuable information the program below provides about the value stored in memory location INPUT. Assume that there is a non-zero value at location INPUT before the program is executed.

HINT: When testing different values of INPUT pay attention to their bit patterns. How does the bit pattern correspond to the RESULT?

```

                                .ORIG x3000
                                LD R0, INPUT
                                AND R3, R3, #0
                                LEA R6, MASKS
                                LD R1, COUNT
LOOP    LDR R2, R6, #0
                                ADD R3, R3, R3
                                AND R5, R0, R2
                                BRz SKIP

```


	ADD R3, R3, #1
	ADD R0, R5, #0
SKIP	ADD R6, R6, #1
	ADD R1, R1, #-1
	BRp LOOP
	ST R3, RESULT
	HALT
COUNT	.FILL #4
MASKS	.FILL 0xFF00
	.FILL 0xF0F0
	.FILL 0xCCCC
	.FILL 0xAAAA
INPUT	.BLKW 1
RESULT	.BLKW 1
	.END

13. Figure out what the following program does.

	.ORIG X3000
	LEA R2, C
	LDR R1, R2, #0
	LDI R6, C
	LDR R5, R1, #-3
	ST R5, C
	LDR R5, R1, #-4
	LDR R0, R2, #1
	JSRR R5
	AND R3, R3, #0
	ADD R3, R3, #7
	LEA R4, B
A	STR R4, R1, #0
	ADD R4, R4, #2
	ADD R1, R1, #1
	ADD R3, R3, #-1
	BRP A
	HALT
B	ADD R2, R2, #1
	LDR R0, R2, #0
	JSRR R5
	TRAP X29

```

ADD R2, R2, #15
ADD R0, R2, #3
LD R5, C
TRAP X2B
ADD R2, R2, #5
LDR R0, R2, #0
JSRR R5
TRAP X27
JSRR R5
JSRR R6
C .FILL X25
.STRINGZ "EE306 and tests are awesome"
.END

```

14. (Adapted from 9.53)

Suppose we want to introduce two extra interrupts to the LC-3: INTA and INTB. INTA has priority 2 and an interrupt vector of x50. INTB has priority 4 and an interrupt vector of x60.

Recall that the priority is specified in bits [10:8] of the PSR. In fact, the full PSR specification is:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSR:	Pr	0	0	0	0	Priority		0	0	0	0	0	0	N	Z	P

where

- PSR[15] = 0 (Supervisor mode), 1 (User mode).
- PSR[14:11] = 0000
- PSR[10:8] = priority, 0 (lowest) to 7 (highest).
- PSR[7:3] = 00000
- PSR[2:0] = condition codes for N,Z,P

We want to provide some flexibility for developers to add their own INTA and INTB service routines, so we currently left them blank as shown below:

INTA service routine:

```

.ORIG x1000
RTI
.END

```

INTB service routine:

```

.ORIG x2000
RTI
.END

```

- a. In order to support INTA and INTB, the interrupt vector table must have entries. Show the addresses of these entries and the contents of those memory locations.

Memory Address	Content

- b. Show how the content of PSR changes after the following user program starts executing at priority 0 and right before the HALT instruction.

```

.ORIG x3000
LD    R0, NUM
INTA occurs
AND   R1, R1, # 0
ADD   R1, R1, # 5
INTB occurs
ADD   R2, R0, R1
HALT
NUM   .FILL xFFF1

```

[illegible]

15. Fill out the control store table for all of the states. If a state does not care about a particular signal, write down x.

You can either build your own table from excel/google sheets.

Empty Sheets: [+ Empty Control Store 306](#)

FSM Example: Extra hard Part-time Job

Input: C, K, N You are a part time worker (for ∞ clock cycles).
Output: P, B, W You get 2 cycles of Break (B) for every 3 cycles you've worked, regardless of whether you were productive (P) or not (W). By default, you are simply working (W), but if you get a compliment (C) from your manager, you will be extra productive (P) in the next 2 clock cycles that you work, or until your next break. (If you get a compliment during your break, ^{or as you're returning from break} you will be productive for the next 2 cycles.) However, if you encounter a Karen (K), which can only happen when you're working, either your mood will be ruined (from P to W for the next cycle) or you will be so stressed out that you go straight to your 2 cycles of break (from W to B). Most of the time, your cycles are Normal (N), meaning nothing significant happens. You start your eternal job by working.

Example:

Input: N N N C N N N C N C K K N...

Output: W W W B B P P W B B P W B...