Lab 3: Task Counter

Professor Patt has a bad memory. He often cannot remember how many tasks left that he has to deal with. As a computer man, he would like to have a screen in front of his desk, reminding the number of tasks all the time. If he finishes one task, he can hit the Enter key excitedly then the task counter decreases by one.

Your task is to implement this task counter by LC-3 interrupt-driven I/O.

> This lab is associated with the following chapters and sections. If you have no idea how to do this lab, please review the corresponding parts of the book.
>
> - 9.1 Privilege, Priority, and the Memory Address Space
> - 9.2 Input/Output
> - 9.4 Interrupts and Interrupt-Driven I/O

Implementation Details

You are required to write in **LC-3 assembly language**. Your program consists of 3 parts:

- The user program, starting at x3000.
- The interrupt service routine, starting at x0800.
- And the system booting code, starting at x0200.

You can write 3 pieces of code in one `Lab3.asm` file, with 3 pairs of `.ORIG` and `.END`. The simulator can load them at the correct locations and put the PC pointing to the first piece of code. The system booting code should be executed at first, so you can organize your code as:

```
        .ORIG x0200
        ...  ; system booting code
        .END

        .ORIG x0800
        ...  ; interrupt service routine
        .END

        .ORIG x3000
        ...  ; user program
        .END
```

Note: The symbol table is shared between program pieces. But if you want to refer a symbol in another program piece, the offset number may be too large for some instructions. In this situation, the assembler says "cannot encode as x-bit 2's complement number". Consider to use different names pointing to the area in the current program piece separately.

User program

The user program prints the number of tasks continuously and infinitely. For every 40 cycles of output, print one newline character. For instance, if the current task counter is 7, it should prints:

```
7777777777777777777777777777777777777777
7777777777777777777777777777777777777777
7777777777777777777777777777777777777777
77777777777777777777777777777...
```

To ensure that the screen does not scroll too fast, you can add a delay subroutine between each time the console prints one character. A simple way to delay is to count down from 256 to 0, like the following:

```
DELAY       ST      R1, DELAY_R1
            LD      R1, DELAY_COUNT
DELAY_LOOP  ADD     R1, R1, #-1
            BRnp    DELAY_LOOP
            LD      R1, DELAY_R1
            RET

DELAY_COUNT .FILL   #256
DELAY_R1    .BLKW   #1
```

In this way, you can simply `JSR DELAY` before each time you print the counter to the screen.

Since 7 is Patt's lucky number, the task counter is initialized as 7.

Interrupt service routine

Once someone hits the keyboard, the keyboard interrupt service should handle the input and do the following instructions:

- If the input is the Enter key, then decrease the task counter in the user program by 1. Specially, if the task counter is already 0 and an Enter key is detected, then ignore the input and do nothing.
- If the input is 0~9, then set the task counter in the user program as the input number.
- Otherwise, echo the input character 40 times in a new line, and return with no change of the task counter.

System booting code

You may have observed that the system booting code in the LC-3 simulator is located at x0200. The original system booting code is executed in supervisor mode. It initializes the system stack at x3000 (`OS_SP`), and then pushes PSR (`USER_PSR`) and PC (`USER_PC`) for user mode, and then finally RTI to the user code at x3000 (`USER_PC`):

```
            .ORIG   x0200
            LD      R6, OS_SP

            LD      R0, USER_PSR
            ADD     R6, R6, #-1
            STR     R0, R6, #0

            LD      R0, USER_PC
            ADD     R6, R6, #-1
            STR     R0, R6, #0

            RTI

OS_SP       .FILL   x3000
USER_PSR    .FILL   x8002
USER_PC     .FILL   x3000
```

In order to make the keyboard interrupt enable, some operations are necessary. Please re-write the system booting code, furnishing the requirements for the keyboard interrupt. Here are some hints:

1. You should enable the keyboard interrupt. The interrupt enable bit is part of the device status register. For the keyboard, the device status register, i.e. KBSR, is located at ________. Therefore, you should do ________.
2. You should let the machine know how to handle the keyboard interrupt. The interrupt vector INTV for the keyboard is x80. Therefore, you should put the starting address of ________ in the correct place in the interrupt vector table, that is ________.

How to Run?

1. Reinitialize or randomize the machine.
2. Load your program into the simulator. Now PC should be x0200, PSR should be x0002 (supervisor mode), and the user program, the interrupt service routine, and the system booting code are all loaded into the memory.
3. Run the machine.
4. Hit on the keyboard and see what happens.
5. If you want to stop the machine, click the Run button again, and the program should be paused.

Note: In the simulator, reinitializing or randomizing will not affect the device registers in xFE00 ~ xFFFF (which the TAs think is a bug). If you have to do so, please shut down the simulator and restart it.

Samples

```
7777777777777777777777777777777777777777
7777777777777777777777777777777777777777
7777777777777777777777777777777777777777
777777777777777777777777777777<Enter>
6666666666666666666666666666666666666666
6666666666666666666666666666666666666666
666666666666666666666666<Enter>
5555555555555555555555555555555555555555
5555555555555555<a>
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
5555555555555555555555555555555555555555
5555555555555555555555555555555555555555
55555555555555555555555555555<9>
9999999999999999999999999999999999999999
9999999999999999999999999999999999999999
999999999999999999<0>
0000000000000000000000000000000000000000
000000000000000000000000000000<Enter>
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
...
```

- `<Enter>`, `<a>`, `<9>`, `<0>` is used to identify the key pressed at that moment. You do not need to print it on your console.
- When a key is pressed, the current line is interrupted. You can print the remaining characters after returning from the interrupt:

  ```
  7777777777777777777777777777777777777777
  7777777777777777777777777777776666666666
  6666666666666666666666666666666666666666
  6666666666666666666666666666666666666666
  ...
  ```

  or just create a new line:

  ```
  7777777777777777777777777777777777777777
  777777777777777777777777777777
  6666666666666666666666666666666666666666
  6666666666666666666666666666666666666666
  ...
  ```

  Both of the solutions are accepted. But you should be aware of why your program performs in that way.

Limitations

- The task counter c: $0 \le c < 10$.

Grading

Lab 3 takes 8% of the total score, consisting of Check part (50%) and Report part (50%).

Check part (50%)

- Find a TA to check your code in person. TAs may ask you questions when grading your lab assignment. You will get 100%, 80% or 60% of the checking score according to your response.
- You can try again if you fail in checking, but there will be a penalty of -10% (of checking part) for each try.
- We suggest you to write enough comments in your code so that you will be aware of what's going on in your program and confident to answer TA's questions.

Report part (50%)

- English report should be concise and carrying main ideas. Try to use the report to convince TAs that you complete the task by yourself.
- Your lab report should *at least* contains the following contents:
    - Your algorithm. To make it clear, you can use figures, tables or any other easy-to-understand appearance.
    - Essential parts of your code with sufficient comments. Please only select the most important code phases and explain them.
    - The questions that TA asked you, and answers.
- 2~4 A4 pages. No template provided. Be sure to make it readable.

Penalty

- **Wrong Answer**: -10% of Check part each time.
- **Delay**: -20% of the corresponding part per day.
- **Cheating**: -100% of this lab. Additionally, -10% of the final score of this course.