LAB 3

Days of the week

3.1 Problem Statement

- Write a program in LC-3 assembly language that keeps prompting for an integer in the range 0-6, and each time it outputs the corresponding name of the day. If a key other than '0' through '6' is pressed, the program exits.

3.1.1 Inputs

At the prompt "**Please enter number:** ," a key is pressed.

3.1.2 Outputs

If the key pressed is '0' through '6', the corresponding name of the day of the week appears on the screen. Precisely, the correspondence is according to this table:

Code	Day
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday

When the day is displayed, the prompt "**Please enter number:** " appears again and the program expects another input. If any key other that '0' through '6' is pressed, the program exits.

3.2 The lab

3.2.1 Strings in LC-3

It will be necessary to define the prompt "**Please enter number:** " and the days of the week as strings in memory. All strings should terminate with the NUL character (ASCII 0). In LC-3 one character per memory location is stored. Each location is 16 bits wide. The 8 most significant bits are 0, while the 8 least significant bits hold the ASCII value of the character. Strings terminated with the NUL character can be conveniently defined using the directive **.STRINGZ "ABC"** , where

"ABC" is any alphanumeric string. It automatically appends the NUL character to the string. As an example, a string defined in assembly language and the corresponding contents of memory are shown in figure 3.1.

```
1    .ORIG x3100
2    .STRINGZ "Sunday"
```

```
x3100 0053 ; S
x3101 0075 ; u
x3102 006e ; n
x3103 0064 ; d
x3104 0061 ; a
x3105 0079 ; y
x3106 0000 ; NUL
```

Figure 3.1: The string "Sunday" in assembly and its corresponding binary representation

3.2.2 How to output a string on the display

To output is a string on the screen, one needs to place the beginning address of the string in register **R0**, and then call the **PUTS** assembly command, which is another name for the instruction **TRAP x22** . For example, to output "ABC", one can do the following:

```
1            LEA R0, ABCLBL ; Loads address of ABC string into R0
2            PUTS
3            ...
4            HALT
5            ...
6 ABCLBL     .STRINGZ "ABC"
7            ...
```

The **PUTS** command calls a system trap routine which outputs the NUL terminated string the address of its first character is found in register **R0**.

3.2.3 How to read an input value

The assembly command **GETC** , which is another name for **TRAP x20** , reads a single character from the keyboard and places its ASCII value in register **R0**. The 8 most significant bits of **R0** are cleared. There is no echo of the read character. For example, one may use the following code to read a single numerical character, 0 through 9, and place its value in register **R3**:

```
1            GETC       ; Place ASCII value of input character into R0
2            ADD R3, R0, x0   ; Copy R0 into R3
3            ADD R3, R3, #-16; Subtract 48, the ASCII value of 0
4            ADD R3, R3, #-16
5            ADD R3, R3, #-16; R3 now contains the actual value
```

Notice that it was necessary to use three instructions to subtract 48, since the maximum possible value of the immediate operand of ADD is 5 bits, in two's complement format. Thus, -16 is the most we can subtract with the immediate version of the **ADD** instruction. As an example, if the pressed key was "5", its ASCII value 53 will be placed in **R0**. Subtracting 48 from 53, the value 5 results, as expected, and is placed in register **R3**.

3.2.4 Defining the days of the week

For ease of programming one may define the days of the week so the they have the same length. We note that "Wednesday" has the largest string length: 9. As a NUL terminated string, it occupies 10 locations in memory. In listing 3.1 define all days so that they have the same length.

```
1            ...
2            HALT
3            ...
4  DAYS      .STRINGZ "Sunday    "
5            .STRINGZ "Monday    "
6            .STRINGZ "Tuesday   "
7            .STRINGZ "Wednesday"
8            .STRINGZ "Thursday  "
9            .STRINGZ "Friday    "
10           .STRINGZ "Saturday  "
```

Listing 3.1: Days of the week data.

If the numerical code for a day is i (a value in the range 0 through 6, see section 7.1.2 on page 7–1), the address of the corresponding day is found by this formula:

$$\text{Address\_of(DAYS)} + i * 10 \tag{3.1}$$

Address_of(DAYS) is the address of label **DAYS**, which is the beginning address of the string "Sunday." Since LC-3 does not provide multiplication, one has to implement it. One can display the day that corresponds to i by means of the code in listing 3.2, which includes the code of listing 3.1. Register **R3** is assumed to contain i.

```
1            ...
2  ; R3 already contains the numerical code of the day i
3            LEA R0, DAYS        ; Address of "Sunday" in R0
4            ADD R3, R3, x0      ; To be able to use condition codes
5  ; The loop (4 instructions) implements  R0 ← R0 + 10 * i
6  LOOP      BRz DISPLAY
7            ADD R0, R0, #10     ; Go to next day
8            ADD R3, R3, #-1     ; Decrement loop variable
9            BR  LOOP
10 DISPLAY   PUTS
11           ...
12           HALT
13           ...
14 DAYS      .STRINGZ "Sunday    "
15           .STRINGZ "Monday    "
16           .STRINGZ "Tuesday   "
17           .STRINGZ "Wednesday"
18           .STRINGZ "Thursday  "
19           .STRINGZ "Friday    "
20           .STRINGZ "Saturday  "
```

Listing 3.2: Display the day.

3.3 Testing

Test the program with all input keys '0' through '6' to make sure the correct day is displayed, and with several keys outside that range, to ascertain that the program terminates.

3.4 What to turn in

- A hardcopy of the assembly source code.
- Electronic version of the assembly code.
- For each of the input $i = 0, 1, 4, 6$, screenshots that show the output.