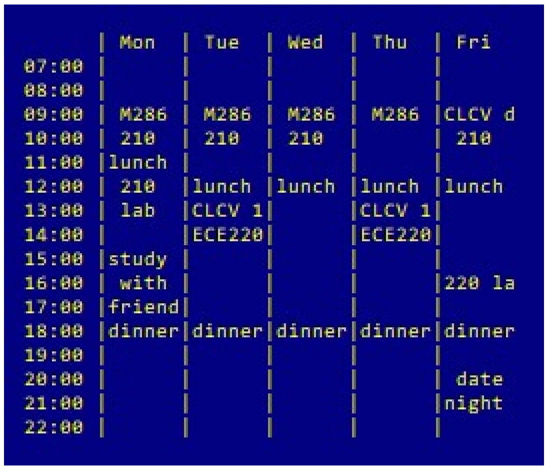
ECE220: Computer Systems & Programming **SPRING 2019 Honors Section**
Machine Problem 1

Printing Subroutines

Your task this week is to write two subroutines to support printing of a student's daily schedule. One subroutine prints an hour, such as "07:00" or "12:00", followed by a space. The second subroutine prints an arbitrary string centered in six spaces, truncating the string or adding extra space characters as necessary. Together, the two subroutines require about 100 lines of LC-3 assembly, including comments.



```
      | Mon  | Tue  | Wed  | Thu  | Fri
07:00 |      |      |      |      |
08:00 |      |      |      |      |
09:00 | M286 | M286 | M286 | M286 |CLCV d
10:00 | 210  | 210  | 210  |      | 210
11:00 |lunch |      |      |      |
12:00 | 210  |lunch |lunch |lunch |lunch
13:00 | lab  |CLCV 1|      |CLCV 1|
14:00 |      |ECE220|      |ECE220|
15:00 |study |      |      |      |
16:00 | with |      |      |      |220 la
17:00 |friend|      |      |      |
18:00 |dinner|dinner|dinner|dinner|dinner
19:00 |      |      |      |      |
20:00 |      |      |      |      | date
21:00 |      |      |      |      |night
22:00 |      |      |      |      |
```

The objective for this week is to give you some experience with decomposing tasks into algorithms, with writing LC-3 assembly code, and with formatting output.

The Task

The first subroutine is PRINT_SLOT. A number from 0 to 15 is passed to your subroutine in R1. The number corresponds to one of the rows of a schedule, as shown in the figure above. Your subroutine must print the time corresponding to the specified slot. If R1=0, for example, your subroutine must print "07:00" followed by a single trailing space (ASCII x20).

The second subroutine is PRINT_CENTERED. A string (the address of the first ASCII character in sequence terminated by an ASCII NUL, x00) is passed to your subroutine in R1. Your subroutine must print exactly six characters. If the string is longer than six characters, your subroutine must print the first six characters. If the string is shorter than six characters, your subroutine must print additional spaces around the string to bring the total length to six characters. If the number of spaces needed is odd, the subroutine must use one more trailing space than leading space.

To make your life easier in the later MPs, neither of these subroutines may change any register values. In other words, except for R7, which is modified by JSR, both subroutines must preserve all bits in all other registers. Note that you are probably going to want to use TRAP instructions in both subroutines, so you should also preserve the return address (in R7) and restore it before executing the RET (JMP R7) instruction.

You may find it useful to develop look-up tables for one or both of these subroutines. A look-up table specifies a function from a small, contiguous set of integers to an arbitrary result type. For example, in a later MP, you might use a lookup table to translate an integer from 0 to 4 into one of the day names at the top of the schedule in the figure ("Mon", "Tue", and so forth). If each element in a look-up table requires a single memory address, one can add the index to the base address for the table to find the address at which the desired element is stored.

You may want to use a lookup table to translate string length into a count of leading spaces, for example.

You may also find it tempting to use a look-up table of strings for slots. While such an approach is acceptable, we encourage you to think about other ways to generate the hour output.

As you develop your subroutines, you may want to include code to test them at the start of the assembly file. Since the LC-3 will begin execution at the start of the file by default, you can then easily test your subroutines with the LC-3 tools.

Specifics

- Your code must be written in LC-3 assembly language and must be contained in a single file called mp1.asm. We **will not grade** files with any other name.
- Your subroutine for printing one of the schedule slot hours must be called PRINT_SLOT.
    - You may assume that R1 (the slot number) holds a value from 0 to 15 when your subroutine is called.
    - Your subroutine must not change any register values other than R7.
    - Your subroutine must output exactly six characters to the display, including an appropriate hour number (R1+7 as a two-digit number, possibly including a leading zero), a colon (:), two zeroes, and a trailing space.
- Your subroutine for printing a string centered in six characters must be called PRINT_CENTERED.
    - You may assume that the string starting at the address in R1 is valid and is terminated by an ASCII NUL character (x00, extended to x0000 in LC-3 memory).
    - You may NOT make any assumptions about string length.
    - Your subroutine must not change any register values other than R7.
    - Your subroutine must output exactly six characters to the display. For strings longer than six characters, the first six characters should be displayed. For strings shorter than six characters, leading and trailing spaces must be added to output six total characters. For short, odd-length strings, use one more trailing space than leading space.
- Your code must be well-commented, and must include a table describing how registers are used within each subroutine. Follow the style of examples provided to you in class and in the textbook.
- You may leave any code that you have used for testing at the start of your file, provided that it does not in any way interfere with your subroutines' functionality.

Testing

We suggest that you write a loop to output all possible values of the slot number for PRINT_SLOT. There are only 16 of them.

We also suggest that you test PRINT_CENTERED with strings of various lengths.

Grading Rubric

Functionality (65%)

- PRINT_SLOT
  - 15% - produces correct output for any slot number from 0 to 15 (in R1)
  - 10% - does not modify any register value other than R7 when called
- PRINT_CENTERED
  - 30% - produces correct output for any string length (string in R1)
  - 10% - does not modify any register value other than R7 when called

Style (20%)

- 10% - PRINT_SLOT not organized as nested conditionals (a look-up table is ok; 16 chunks of code selected by conditionals are not)
- 10% - PRINT_CENTERED not organized as nested conditionals (use a lookup table or a right shift)

Comments, Clarity, and Write-up (15%)

- 5% - each subroutine has a paragraph explaining what it does and how it must be called (these are given to you; you just need to document your work)
- 10% - code is clear and well-commented

Note that some categories in the rubric may depend on other categories and/or criteria. For example, if your code does not assemble, you will receive no functionality points. Note also that the remaining LC-3 MPs (two of them) will build on these subroutines, so you may have difficulty testing those MPs if your code does not work properly for this MP.