COE 205 Computer Organization & Assembly Language – Fall 2005

Assignment 5: Bit Manipulation Operations

Professor: Muhamed Mudawar

Due Date: Saturday, December 3, 2005

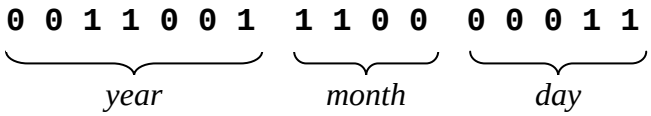
Q1. (10 pts) Write an assembly language program to read a 32-bit hexadecimal number and does the following operations. A sample run is shown below:

- Using only shift instructions, convert the hexadecimal number (in ASCII) to binary.
- Display the 32-bit number in binary (each bit is converted to ASCII before displaying it)
- Count and display the number of 1's in the binary representation
- Count and display longest sequence of 0's

```
Enter number in hex (max 8 digits): 7e1A3C0f
The binary equivalent is: 01111110 00011010 00111100 00001111
Count of 1's in binary representation: 17
Longest sequence of 0's: 6
Repeat program (Y/N)? n
```

Read the input character by character. Make sure to validate user input. For example, if the user enters any character other than **0** thru **9**, **A** thru **F**, and **a** thru **f**, then do not echo and do not accept the character. Similarly, the program should only accept **Y**, **y**, **N**, and **n** as valid answers for repeating. Other characters should not be echoed nor accepted. Insert a space character between the bytes to enhance the readability of the binary number.

Q2. (10 pts) Write an assembly language program to read a 4-digit hexadecimal number from the user, representing a **date stamp** and outputs the corresponding date in a readable format. The format of the date stamp is as follows:



The least significant 5 bits represent the day number, the next 4 bits represent the month number, and the most significant 7 bits represent the year number relative to 1980. Thus, the above day number is 3, the month number is 12 (December), and the year number is 25 + 1980 = 2005. The above date should be displayed as 3-DEC-2005. Here is a sample run:

```
Enter date stamp as 4 hex digits: 3383
Date is: 3-DEC-2005
Repeat program (Y/N)? n
```

Read the input character by character as in the first question and filter the keyboard so that it does not accept an invalid character as input. Validate the day number and the month number after reading them. Day 0 and months 0, 13, 14, and 15 should not be accepted, and an error message: **INVALID DATE** should be displayed. The following month names should be used (JAN=1, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC=12).

You can get extra marks on this assignment beyond the full mark if you improve your program to include the following:

First Bonus = 1 pt (day number is out of range)

Check whether the day number is out of range for a given month. For example, there are exactly 30 days in April (month 4), so April 31 is not allowed. Similarly, you can't have more than 29 days in February (month 2), so February 30 and 31 are not allowed.

Second Bonus = 1 pt (day number is out of range for February)

The month of February is typically 28 days, but can be 29 days in a leap year. A leap year is a year divisible by 4, so 1980 is a leap year, but 1981 is not. So, February 29, 1980 is allowed, but February 29, 1981 is NOT allowed. Check whether the day number is out of range for February.

Third Bonus = 2 pts (Compute the weekday from the date)

1-JAN-1980 is a Tuesday and can be displayed as TUE 1-JAN-1980. Given a valid date, compute and display the weekday from the date. Here, you need to compute the number of days since 1-JAN-1980 and then compute the weekday based on that reference date. If done properly, you should get Saturday for 3-DEC-2005, displayed as SAT 3-DEC-2005.

Documentation and Grading

Make sure to document your code and make it as readable as possible. 20% of the mark will go to documentation and readability, 30% will go to the code, and 50% will go to correctness. So, a non-running program can receive at most 50% of the grade. Write your name, your id, the date, the objective, the input, and the output at the beginning of each program.

Submitting Programming Assignments

- **Submit the source and executable files of each program.** A missing executable file will receive 0 on correctness. You may put all files in one zip file or attach them separately.
- **All submissions should be made through WebCT on the due date by 11 pm.**
- **Late programming assignments will be accepted, but 10% of the grade will be deducted for each late day for a maximum of 5 late days**. Assignments submitted after 5 late days will NOT be graded.
- **A program can be submitted ONCE. Multiple submissions are NOT allowed.** So, make sure to test your program fully using many inputs before submitting it. A small programming error might cost you a lot in program correctness. If your program is not running properly, then consider fixing it and submitting it late by one day losing only 10% of the grade rather than submitting it incorrectly and loosing up to 50% on correctness.
- **Cheating on programming assignments will NOT be tolerated.** All detected cheating cases will receive zeros, including those students who made the effort and wrote the program. So, make sure that you do NOT give a copy of your program to your friends, because then you might loose your mark.