CSC201, SECTION 002, Fall 2000: Homework Assignment #1

DUE DATE

Monday, September 11, at the start of class.

INSTRUCTIONS FOR PREPARATION

- Neat, in order, answers easy to find.
- Staple the pages together at the upper left corner.
- Fold lengthwise. On the outside write the course number, the assignment number, the date, and your name.

Thanks for your help.

PROBLEMS

1. (2)
    1. If someone tells you about a fancy new set of instructions that their processor implements, are they talking about computer organization, architecture, or design?
    2. If someone shows you a detailed diagram of how the ALU on their processor is built, are they talking about computer organization, architecture, or design?
2. (4)
    1. If processor performance keeps improving at a rate of 50% / year, how long will it take before processors are 10 times more powerful than they are today?
    2. If disk prices keep dropping by 40% / year and a gigabyte costs approximately $10 today, how long will it be before a gigabyte costs $1?
3. (4) How much faster is a Pentium II than the 80386 of 10 years ago (based purely on clock speed), and how much more complex is a Pentium II than an 80386 (based purely on transistor count)?
4. (2) Why do you think there are there multiple buses in a computer system (i.e., why isn't there just one set of wires that connects everything together)?

5. (6) If a memory system is byte-addressable, and the memory has *n* address lines, how big (in bytes) is the maximum addressable size of the memory if *n* is:
    1. 16
    2. 24
    3. 32
6. (2) What is an example of something that might be stored in a ROM of a computer system?
7. (4)
    1. Suppose the instruction cycle takes 5 ticks of the clock, 1 tick for each of the 5 steps. How long does it take to execute 100 instructions if they are not overlapped (i.e., one instruction goes through all 5 steps before the next instruction is begun)?
    2. Suppose instruction execution is overlapped, so that every clock tick a new instruction is started (i.e., instruction 1 begins on the first clock tick and ends on the fifth clock tick; instruction 2 begins on the second clock tick and ends on the sixth clock tick; etc.). How many clock ticks does it take to execute 100 instructions to completion under this assumption?
8. (6) Below is a SASM program with five errors (syntax errors, or necessary things that are missing). Identify the errors.

```
.486                              /* use the Intel 486 instruction set */
.model  flat, stdcall

.stack  1000h

include sasmacros.inc             ;;; this file defines the SASM instructions
title  hello  program

.data
num1       dd   1234h
num2       dd   5678
string1    db   Hello, world!, 0ah, 0dh, 0

.code
          add     num1, num2
          put_i    num1
;         put_i    num2
          put_str  string1
end
```

9. (6) Which of the following are reserved words which you should not use as labels in your programs?
    1. bnqz
    2. bx
    3. bound
    4. else
    5. groups
    6. if
    7. mom
    8. pop
    9. rash

10. sp2
11. while
12. whynot

10. (8) Write the data definition statements (DB or DD) that define the following variables:
    1. a byte-sized integer labeled "var1" with no specific initial value
    2. a doubleword-sized integer labeled "var2" with initial value of -258
    3. a doubleword-sized floating-point (real) number labeled "var3" with value -.02681
    4. a null-terminated character string labeled "var4" with value equal to the last 4 letters of the alphabet (all lower case; do not include any carriage return or newline characters)
    5. a null-terminated character string labeled "var5" with value equal to the digits 1 through 3, followed by the newline (0ah) and carriage return (odh) characters, followed by the digits 4 through 8, followed by the carriage return and newline characters (and then null-terminated)
    6. an array of 8 byte-sized integers labeled "arr1", all initialized to 0
    7. an array of 100 character strings, each of length 5, and each with value "aBcD", 0 (i.e., each is null terminated)

11. (4) For each of the following questions, assume that a, b, and c are doubleword-sized integers, with initial values 10, 25, and -4 (respectively)
    1. what is the value of b after executing the instruction "mov b, c"?
    2. what is the value of a after executing the instruction "iadd b, a"?
    3. what is the value of b after executing the instruction "irem b, c"?

12. (6) Which of the following are "legal" arithmetic instructions? Assume that a and b are doubleword-sized integers, while x and y are doubleword-sized floating point numbers.
    1. iadd a, 25
    2. isub b, 3.14159
    3. fpmult x, 35.1e-8
    4. ineg b
    5. fpadd x, a
    6. imult 25, a

13. (4) What value will the condition codes (SF and ZF) have after executing each of the instructions below? Assume for each problem that the doubleword-sized integers g and h are initialized to 35 and 6, respectively.
    1. iadd g, h
    2. isub h, h
    3. compare h, g
    4. irem h, g

14. (4) What value will a have after executing the instruction "irem a, b", if the doubleword-sized integers a and b have the initial values shown?
    1. a = 100, b = 7
    2. a = -100, b = -7

15. (6) For each of the expressions shown below, write the code to evaluate the expression. All variables are doubleword-sized integers.
    1. a + (b - c)
    2. (w * x) + (y / z)
    3. e + f * g - 3
16. (6) What will be the result of executing each of the following instructions? Assume the variables are initialized before each problem as follows:

    ```
    a    db    ?
    b    db    'b'
    c    db    '001234', 0ah, 0dh, 0
    d    dw    001234
    x    dd    3.14159
    ```

    1. get_ch a
    2. put_ch c
    3. put_str c
    4. put_fp x
    5. put_i c
    6. put_i d
17. (8) Write the code to implement the following C language statements. Assume all variables are doubleword-sized integers, except for j and k, which are byte-sized integers.
    1. `if (a == b) c = 3;`
    2. `if (j < k) d = e;`
    3. `if ((w > x) || (y == z)) x = x + 1;`
    4. `if ((z < 3) && !(y == 5)) z = x - y;`
18. (6) Write the SASM code to implement the following C language statements. Assume all variables are doubleword-sized integers.

    1.
       ```
       if (a == b)
           c = 2;
       else
           c = 3;
       ```

    2.
       ```
       if (x == y)
           c = 2;
       else if (w < z)
           c = 3;
       ```
19. (6) Write the SASM code to implement the following C language statements. Assume all variables are doubleword-sized integers.

    1.
       ```
       while (1) {
           a = a + 1;
           c = c / 2;
       }
       ```

    2.
       ```
       while (a > 15) {
           d = d + e;
       ```

```
          a = a - d;
      }
```

20. (6) Write the SASM code to implement the following C language statements. Assume all variables are doubleword-sized integers.

    1.
       ```
       for (i = 0; i < 10; i++) {
            putc(c);
       }
       ```

    2.
       ```
       for (j = b; j > 0; j = j / 2) {
            a = a * 3;
       }
       ```

PROGRAMMING

In class we will review how to use the Microsoft programming tools. Things to remember:

- Create a directory with nothing in it except your Assembly Language program file.
- Copy the sample makefile, and sasmacros.inc, into this directory.
- Suppose you call this file "hw1a.asm". Do NOT name your makefile "hw1a.mak"; name it something else, like "test.mak". You will need to edit the makefile so that all references to a program name are "hw1a". It's also not a bad idea to give your program the title "hw1a" in the assembly language file.
- Start up Visual C/C++. Once running Visual C++, you want to OPEN the file test.mak (or whatever you called your makefile). You'll need to find the directory (folder) where you put the files. In the right directory, list files of type All Files, and open as type AUTO (the default). You will get a dialog box that says something to the point that this make file was not generated by the Developer's Studio, and continuing will "wrap" the makefile. You want to "wrap" this make file, so click on YES. You will get another dialog box that assigns the Win32 platform. Click on OK.
- When it asks what you want to save this as, type "hw1a.dsp".
- You are now ready to work on the assembly language program. If you want, OPEN the file hw1a.asm as type AUTO (the default), to see what is in it.
- Before you can execute (run) a program, you need to assemble and link the program. (This is similar to when you compiled a C++ program.) Your makefile together with the Visual C++ tool will do this for you. To do this, under BUILD, do either BUILD ALL or BUILD hw1a.exe. You will see a window pop up that tells you what is being built, and if there were errors, you would see them here.
- If there were errors, then you would iterate on the steps of modifying your source code (the file with the .asm extension), save the file, and again BUILD, until your program makes it through the build with no errors and no warnings.
- At this point, you can execute the program. Do this by choosing EXECUTE hw1a.exe under the BUILD menu.
- I encourage you to use the source debugging tool to see what your program is doing during execution. You may choose instead simply to use print statements to output the state of variables in your program; in this case, you do not need to use Visual Studio for executing

or debugging your program.

1. (50) Write a SASM program (named "hw1a.asm") that outputs the message "Enter two 2-digit decimal numbers, each followed by a carriage return", then allows a user to type in two numbers and outputs the message

   ```
       A is greater than B
   ```

   where A is the larger of the two numbers. Some additional constraints:
    - each of the two numbers entered by the user will be two decimal digits long, followed by a carriage return typed by the user.
    - Any messages output by your program should end with newline and carriage return characters (0ah and 0dh, respectively).
    - Your program ends normally after outputting the messages shown above.

   I will give instructions for submitting your program electronically at a later time.

   WARNING! Keep all your program files strictly in your EOS locker space, with no access given to other users. Do not put your files on the local machine, such as in temp space! If you do, you are inviting someone else to use your code, since files on the local PC file system are not protected.

GRADING

This homework is graded on a scale of 150 points; the points for each problem are shown above. The homework score will be weighted to contribute 5% of your course grade.

Problems will be graded according to the following interpretation:

- All right = full credit
- halfway to mostly right = half credit
- Not much or none right = no credit