CSC201, SECTION 002, Fall 2000: Homework Assignment #4 Problems

---

DUE DATE

November 15 for the homework problems, in class.

INSTRUCTIONS FOR HOMEWORK PROBLEMS

- 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. (5 points)
   Write the Pentium floating-point code to evaluate the following expressions. Include any data definitions that your code needs. Assume "var1" through "var6" have already been defined as doubleword-length floating-point values.
   1. `resultvar = var1 * (var2 + var3) / var4`
   2. `resultvar = -var1 * (3.1415926 / var4) + var3 - (var5 + var2)`
2. (3 points)
   Write the Pentium floating point code for the following C code. Include all data definitions that your code needs.

```
float d1, d2, d3, d4;

d4 = d2+d3;
if (d4 > d1)
        putchar('g');
else if (d4 < d1)
        putchar('l');
else
```

```
putchar('e');
```

3. (1 points)
   Create a symbolic label "NUMBYTES" for the value of the expression 13*10*2+6*2

4. (1 points)
   Define a macro named SAVEREGS that will always save registers AX through DX on the stack.

5. (2 points)
   Define a macro named STARTPROC that will save EBP on the stack, set EBP equal to ESP, and save a specific set of registers on the stack. The set of registers to be saved is passed as parameters to the macro.
   Show how your macro is called, also.

6. (3 points)
   Define a macro called STRDEF that will define a string with an initial value that is passed as a parameter. The name of the string is "stringXXX", where the value for XXX is also passed as a parameter to the macro. (Ex., the parameters passed to the macro might be "abcdefg", and Alpha6. The result is the statement

   ```
   stringAlpha6  db  "abcdefg", 0
   ```

   Show how your macro is called, also.

7. (3 points)
   Create a macro named ADDTWO that accepts two doubleword parameters (which are both variables in memory), adds the second to the first (storing the result in the first), and then sets the first to zero if the sum is less than zero.

   Show how your macro is called, also.

8. (1 points)
   Create a macro named DEBUGPRINT that has one parameter, a register. This register contains the address of a null-terminated string in memory. If the DEBUG flag is defined, the string should be printed (output) by the macro, otherwise the macro does nothing.

9. (3 points)
   Show the Pentium assembly language code to call a procedure named "addtwo". Procedure addtwo has two input doubleword input parameters, and returns a doubleword value. The inputs to your call are myvar1 and myvar2, which have been defined already. Use the calling conventions discussed in class.

10. (5 points)
    Show the Pentium assembly language code for procedure "addtwo", which adds two

doubleword inputs, and returns the absolute value of the resulting sum. Use the calling conventions discussed in class.

11. (3 points)
    Convert your calling code and addtwo procedure to use registers for passing inputs, local storage, and returning values (i.e., ignore most of our procedure calling conventions). The only requirement for restoring registers is that your procedure should not affect the value of EBP, ESI, or EDI upon return, and of course the stack must be "cleaned up" properly. Show both the calling code and the procedure.

12. (5 points)
    Procedure X calls procedure Y with 2 input parameters. Procedure Y allocates space for 3 local variables, and calls procedure Z with 1 input parameter. Procedure Z returns a single value, as does procedure Y. All parameters and return values are doublewords. Assume that Procedure Y modifies EBP, EAX, and EBX, and procedure Z modifies EBP, EBX, and ESI. Show the contents of the stack in the middle of executing procedure Z (after all the "stuff up front" has been executed, but before any of the "stuff at the end" of a normal procedure is executed). Use the procedure calling conventions from class, and show the current location pointed to by EBP and ESP.

13. (3 points)
    What's wrong with the following procedure (may be multiple problems)?

```
mysub proc
        push    EBP
        mov     ECX, 0
loopit:
        cmp     ECX, [EBP+8]
        je      exit1
        get_ch
        add     AL, 23
        put_ch
        inc     ECX
        jmp     loop1
exit1:
        mov     [EBP-12], ECX
        pop     EAX
        pop     ECX
        pop     EBP
        ret     4
```

14. (2 points)
    1. In the program in figure 12.4 of the text, what statement allocates space for local variables?
    2. Why are _squares and _counter defined as equates (constant values), rather than as statically defined variables (e.g., "_squares dd ?")?

15. (2 points)
    What is the 3-bit encoding for registers
    1. ECX

2. ESI

16. (10 points)
    Show the binary encoding of the following instructions. Variable "myvar" is at address 00010203h.
    1. lea EBX, myvar
    2. dec ESI
    3. cmp EAX, 33224400h
    4. or EDX, EBX
    5. mov EAX, 25h[EDI][EDX]

17. (6 points)
    Show the assembly-language instructions that generated the following binary code.
    1. 59h
    2. 74EDh
    3. 8B4B02h

18. (1 points)
    If instruction "JE PRTERROR" is at address 00400100h and the target of the jump is at address 00400112h, what is the code displacement to the target (Be careful!)?

19. (1 points)
    Module 1 (i.e., file 1) references the variable "myvarx" in module 2, and the code in module 2 contains a jump to the label "printerror" in module 1. Show the statements that must be added to modules 1 and 2 so that linking is done properly.

GRADING

This homework is graded on a scale of 100 points; the points for each problem are shown above. The homework problems are worth 60 points, and the program is worth 40 points. This homework 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