

EECS 370 Winter 2009

Homework 2

Assigned: Tuesday, January 27, 2009

Due: Tuesday, February 10, 2008 (in class)

Name: _____ Uniqname: _____

Instructions:

1. Please write your name and uniqname in the spaces provided above. *Attach this cover sheet* to your completed solutions to the problems listed and turn them in at class on the due date. Submissions without a completed, attached cover sheet cannot be graded.
2. Your answers should be neat, clear, and concise. Computer-written work is recommended (but not required). Show all your work, and state any special or non-obvious assumptions you make.
3. You may discuss your solution methods or your answers with other students, but the solutions you submit must be your own.

Scores:

Problem #	Points
1	/15
2	/8
3	/6
4	/12
5	/20
Total	/50

Q1. (15 points) You are using the “SooperCompiler” to compile the three functions below. function1 is called from main once, while function2 and function3 are both called from function1 only. We report below the relevant snippets of the functions’ code, including all the calls to other functions. The architecture that theSooperCompiler is targeting has 2 caller-saved registers and 2 callee-saved registers

```
int function1(int arnold){
    int a,b,c,d ;
    a = 15 ;
    b = 12 ;
    function2(a)
    c = a + b ;
    while(a > 0) {
        //iterates 15 times
        function3(a) ;
        a-- ;
    }
    d = 9 ;
    d = 4 + arnold ;
    d++ ;
    return 0 ;
}

int function2(int gerald){
    int k,l,m ;
    k = 21 ;
    m = 23 ;
    printf(“ps118\n”) ;
    l = k + m ;
    l = l / 2 ;
    return 0 ;
}

int function3(int helga){
    int w,x,y ;
    w = 4 ;
    x = 9 ;
    y = w + x ;
    printf(“footballhead\n”) ;
    y = w ;
    return 0 ;
}
```

The SooperCompiler has mapped the local variables of the functions as follows:

function1: a and b are in caller-saved, while c and d are in callee-saved

function2: k and l are in caller-saved, while m is in callee-saved

function3: w and y are in caller-saved, while x is in callee-saved

- (a) How many save/restore executed instruction pairs did the compiler have to include in the assembly code of function1 ONLY for the purpose of preserving register values across functioncalls? (6 points)
- (b) Can you do better than the SooperCompiler? Provide the best assignment to caller or callee-saved registers on a per-register basis. If caller and callee-save give the same performance for a variable, then put “either” for your answer. Be sure to state the total number of save/restore instruction pairs added to function1’s assembly code. (9 points)

Q2. (8 points) Given the following C file:

doug.c:

```
extern char a[] ;
```

```
int *b ;
```

```
float c[20] ;
```

```
struct {
```

```
    int d ;
```

```
    char e ;
```

```
} f ;
```

```
extern float g ;
```

```
int skeeter(int) ;
```

```
float porkchop(string h) {
```

```
    float i = 3.14 ;
```

```
    float j = patty(h) ;
```

```
    float k = roger(h) ;
```

```
    static float m = j + k ;
```

```
    return m ;
```

```
}
```

Circle all the symbols that are placed in the symbol table of doug.o:

a b c d e f g h i j k m
skeeter porkchop patty roger

Q3. (6 points) C to MIPS assembly compilation

Following is the code snippet for performing matrix addition. A matrix is simply stored as an array in memory. Assume that the matrices a, b and c start from memory locations $(1000)_{16}$, $(1500)_{16}$, and $(2000)_{16}$ respectively. You are required to write the equivalent MIPS assembly code for the code snippet provided. You are free to use any register available for data operations, provided it is not reserved. Be sure to comment every line of your code for partial credit (and ease in debugging).

Code snippet:

```
int a[10], b[10], c[10];
```

```
int i;
```

```
for(i=0; i<10; i++)
```

```
    c[i] = a[i] + b[i];
```

Q4. (12 points)

Consider assigning the variables below to memory locations starting at address 100 (in decimal). The alignment rules are as discussed in class. The size of each data type is listed below.

```
short a;
double b[10];
short c;
struct {
    char d[5];
    double* e;
    short f;
} g[3];
int z;
```

Data Type	Size
char	1 byte
short	2 bytes
int <i>or</i> pointer	4 bytes
float	4 bytes
double	8 bytes

- (a) Calculate the size and starting address for each variable. Show your work by drawing the memory layout. (8 points)
- (b) How many bytes are required to layout these declarations in memory? (2 points)
- (c) Is it possible to reduce the size of the structure? Explain your reasoning. (2 points)

Q5. (20 points)

CAEN is planning to install a new capacity monitor for the computing labs on campus. This monitor will contain a single light, which will be yellow, orange, or red depending on how many computers are occupied. Due to CAEN's lack of funding, however, this light will be operated manually using two switches, one which controls the yellow LED's, and one which controls the red LED's. Each switch has two positions: "on" and "off". Turning either switch "on" while the other switch is "off" will cause its respective color to be displayed, while turning both switches "on" will display orange. There is also a switch for the light itself, which simply turns the light "on" or "off".

You will analyze and design a state machine for the monitor light in the following series of steps. The possible inputs are (6 total):

- Flip yellow switch on or off
- Flip red switch on or off
- Flip light switch on or off

The possible outputs are (4 total):

Red/orange/yellow light
No light

You may ignore vacuous transitions, such as attempting to flip a switch on when it is already on. Assume that only one switch may be flipped at a time, and that the initial position of all the switches is "off".

- a. (8pt) Draw a state diagram (with input and output values labeled on all arcs)
- b. (6pt) Provide the state table (with state encoding)
- c. (4pt) Determine the output functions (write the corresponding equations); you may omit "no light" as an output
- d. (2pt) Is this a Mealy or Moore machine?