The UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Comp 411 Computer Organization
Spring 2007

Problem Set #3
Issued Monday, 5 February 2007; Due Thursday, 15 February 5, 2007

Homework Information: Some of the problems are probably too long to be done the night before the due date, so plan accordingly. Late homework will not be accepted. Feel free to get help from others, but the work you hand in should be your own.

Problem 1. "Compiler Appreciation"

Translate the following code fragments (written in C) into MIPS assembly language. Use the general approach shown in lecture (allocate variables into low, directly addressable, memory addresses). You don't have to write optimized assembly language unless you are into that sort of thing. Comment your assembly code to show the correspondence between C-language and assembly-language constructs. Just show the executable code—you can assume that the necessary storage allocation for each variable or array has already been done, meaning that a label has been defined for each variable and the first entry of each array. Furthermore, you can assume that all variables have been allocated into the lower 32K bytes of memory, and that all variables and arrays are C integers, i.e., 32-bit values.

(A)
```
y = y + x;
```

(B)
```
a[i] = a[i-1] + a[i-2];
```

(C)
```
if (x > y)
       x = x - y;
   else
       x = y - x;
```

(D)
```
while (j != 0)
        j = j << 2;
```

(E)
```
for (i = 9; i >= 0; i = i – 1)
       a[i] = i + i + 1;
```

(F)
```
a[a[x]] = a[x];
```

Problem 2. "MIPS Calisthenics"

Write MIPS code fragments to perform the following simple tasks.

(A) Clear registers t0-t7
(B) Clear memory locations (words) 0x100 to 0x1fc, inclusive
(C) Swap the contents of registers $t0 and $t1

(D) Count the number of memory locations with zero contents

Problem 3. "Name that Loop"

(A) Describe in your own words the function performed by the following code fragment when it reaches the label `end`. Consider the following hints. The fragment operates on the contents of register a0, and the loop will always complete.

```
          add   $t0,$0,$0
loop:     andi  $t1,$a0,1
          add   $t0,$t0,$t1
          srl   $a0,$a0,1
          bne   $a0,$0,loop
end:      add   $a0,$t0,$0
```

(B) Describe how the operation of the above code fragment changes if the `srl` instruction is replaced with an `sra` instruction with the same arguments.

(C) Describe the function of the following code fragment in your own words:

```
          la    $t1,a
loop:     lw    $t0,4($t1)
          sw    $t0,($t1)
          addi  $t1,$t1,4
          bne   $t0,$0,loop
```

(D) Describe the function of the following code fragment in your own words:

```
          la    $t1,a
loop:     lw    $t0,($t1)
          addi  $t1,$t1,4
          bne   $t0,$0,loop
          la    $t0,a
          sub   $t0,$t1,$t0
          sra   $t0,$t0,2
```

Problem 4. "A Loop of Your Own"

Download SPIM the MIPS instruction set simulator. Then, write your own code fragment to count the number of odd integers in the following array.

a[10] = { -3, 17, 13, 101, -51, 42, 17, 2, 0, -4 };

Your code fragment will begin execution at the label "main", and should terminate with the instruction "j $31". Test your code using SPIM and turn in a copy of the assembly language file and a screen dump of SPIM after the program completes (just before the "j $31" instruction) with the result, which should be stored in some register, highlighted.