CENG 3420 Homework 1

Due: Feb. 18, 2016

Q1 The following problems explore number conversions between binary numbers to decimal numbers.

$1000, 1111, 1011, 1010_{two}$

1. For the patterns above, what base 10 number does the binary number represent, assuming that it is a two's complement integer?
2. For the patterns above, what base 10 number does the binary number represent, assuming that it is an unsigned integer?
3. For the patterns above, what hexadecimal number does it represent?

-57_{ten}

4. For the base ten numbers above, convert to 2's complement binary (16bit).
5. For the base ten numbers above, convert to 2's complement hexadecimal (16bit).

Q2 Consider three different processors P1, P2, and P3 executing the same instruction set with the clock rates and CPIs given in the following table.

Processor	Clock Rate	CPI
P1	2 GHz	1.2
P2	3 GHz	0.8
P3	4 GHz	2.0

1. Which processor has the highest performance expressed in instructions per second?
2. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.
3. We are trying to reduce the time by 30% but this leads an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

Q3 Suppose we have developed new versions of a processor with the following characteristics.

Version	Voltage	Clock Rate
Version 1	1.1 V	3 GHz
Version 2	0.8V	4 GHz

1. How much has the capacitive load varied between versions if the dynamic power of version 2 is 24% less than version 1?
2. How much has the dynamic power been reduced if the capacitive load does not change?
3. Assuming that the capacitive load of version 2 is 80% the capacitive load of version 1, find the voltage for version 2 if the dynamic power of version 2 is reduced by 20% from version 1.

Q4 The following problems deal with translating from C to MIPS. Assume that the variables f , g , h , i , and j are assigned to registers \$s0, \$s1, \$s2, \$s3, and \$s4, respectively. Assume that the base address of the arrays A and B are in registers \$s6 and \$s7 , respectively.

```
f = -g - A[6];
```

1. For the C statements above, what is the corresponding MIPS assembly code?
2. For the C statements above, how many different registers are needed to carry out the C statement?

Q5 The following problems deal with translating from MIPS to C. Assume that the variables f , g , h , i , and j are assigned to registers \$s0 , \$s1 , \$s2 , \$s3 , and \$s4, respectively. Assume that the base address of the arrays A and B are in registers \$s6 and \$s7 , respectively.

```
sll $s2, $s4, 4
add $s0, $s2, $s3
add $s0, $s0, $s1
```

1. For the MIPS assembly instructions above, what is the corresponding C statement?
2. For the MIPS assembly instructions above, rewrite the assembly code to minimize the number if MIPS instructions (if possible) needed to carry out the same function.
3. How many registers are needed to carry out the MIPS assembly as written above? If you could rewrite the code above, what is the minimal number of registers needed?

Q6 Assume that the stack and the static data segments are empty and that the stack and global pointers start at address 0x7fff fffc and 0x1000 8000, respectively. Assume the calling conventions as specified in Figure 2.11 in the textbook and that function inputs are passed using registers \$a0 - \$a3 and returned in register \$v0 . Assume that leaf functions may only use saved registers.

```
int my_global = 100;
main()
{
    int a = 10;
    int b = 20;
    int c = 30;
    int d;
    d = my_function(a,b,c)
```

```
}
int my_function(int a, int b, int c)
{
    return a - b + c + my_global;
}
```

1. Write MIPS assembly code for the code in the table above.
2. Show the contents of the stack and the static data segments after each function call.
3. If the leaf function could use temporary registers ($t0, $t1, *etc.*), write the MIPS code for the code in the table above.

Q7 Show the single MIPS instruction or minimal sequence of instructions for this C statement: `b = 25 | a`; Assume that *a* corresponds to register $t0 and *b* corresponds to register $t1.

Q8 Write a procedure, *findk*, in MIPS assembly language. The procedure should take a single argument that is a pointer to a null-terminated string in register $a0. The *findk* procedure should locate the first 'k' character in the string and return its address in register $v0. If there is no 'k' in the string, then *findk* should return a pointer to the null character at the end of the string. For example, if the argument to *findk* points to the string "hijklmn", then the return value will be a pointer to the fourth character of the string.