CSCI 401 Test 1

Dr. Keith Evan Schubert

1. You are the project lead at Micro Performance. The company wants you to select which option would be best for their new Micro Speed processor.

   (a) Speed up clock

Type	Original	Fast Clock	Percent
Arithmetic	1	1	40%
Branch	2	2	20%
Memory	4	8	40%
Clock	1.8GHz	3.6GHz	

   (b) Speed up memory.
   If you keep the clock the same and add a high performance bus and larger multilevel cache you can improve memory performance by 4 times.

   Justify your answer and specify the performance increase over the original by:

   i Calculate the speedup of the fast clock system over the original.

   ii Calculate the fraction of **time** spent in memory instructions for the original system.

   iii Calculate the speedup of the fast memory system over the original.

   iv Clearly state your choice.

   You don't need a calculator to do the computations, they work out nicely.

$$P_a = \frac{.4*1+.2*2+.4*4}{.4*1+.2*2+.4*8}\frac{3.6GHz}{1.8GHz} \tag{1}$$
$$= \frac{.4+.4+1.6}{.4+.4+3.2}2 \tag{2}$$
$$= \frac{2.4}{4}2 \tag{3}$$
$$= \frac{4.8}{4} \tag{4}$$
$$= 1.2 \tag{5}$$

$$f = \frac{.4*4}{.4*1+.2*2+.4*4} \tag{6}$$
$$= \frac{1.6}{2.4} \tag{7}$$
$$= \frac{2}{3} \tag{8}$$

$$
\begin{aligned}
P_b &= \frac{1}{1/3 + \frac{2/3}{4}} && (9)\\
&= \frac{1}{1/3 + \frac{1}{6}} && (10)\\
&= \frac{1}{\frac{1}{2}} && (11)\\
&= 2 && (12)
\end{aligned}
$$

Improve the memory system for the next processor.

2. Write the MIPS assembly code for the following function. Assume the array a has been defined as size **n**. The following registers are to be used to pass the values:

pointer to a	`$a0`
n	`$a1`
sum	`$v0`

You do not need to write the code to call the function.

```
int sum(int* a, int n){
  int sum;
  sum=0;
  for(int i=0;i<n;i++){
    sum+=a[i]}
  return sum;}
```

```
sum:
  add $v0, $zero, $zero   # sum=0
  sll $a1, $a1, 2         # 4*n
  add $a1, $a1, $a0       # one element after last in array
  ble $a1, $a0, sum_done  # array empty
sum_loop:
  lw $t0, 0($a0)          # get element
  addi $a0, $a0, 4        # increment pointer
  add $v0, $v0, $t0       # add element to sum
  bne $a0, $a1, sum_loop  # check if more elements
sum_done:
  jr $ra                  # return
```

3. Perform the indicated calculations by the algorithm requested showing all steps. Show how you get the number.

(a) 3 + 5 by conditional sum for 4 bit numbers. Assuming the numbers are 2's complement, does overflow occur.

0	0	1	1
0	1	0	1
(01,00)	(10,01)	(10,01)	(11,10)

$\searrow$ $\swarrow$ $\qquad$ $\searrow$ $\swarrow$

(010,001) $\qquad$ (101,100)

$\searrow$ $\swarrow$

(01001,01000)

Since this was addition we take the one on the right and get a carry out of zero (leftmost bit), and an answer of $1000_{2's\ comp} = -8_{10}$. Since we added two positives and got a negative, there was overflow.

(b) 7×-7 by booth's algorithm for 4 bit numbers.

U	V	X	x_{-1}	Comment
0000	0000	0111	0	Setup
0111				subtract -7
0011	1000	1011	1	shift right
0001	1100	1101	1	shift right
0000	1110	1110	1	shift right
1001				add -7
1100	1111	0111	0	shift right and finish

Check: $11001111_{2's\ comp} = -00110001_2 = -49_{10}$.

(c) Convert 19.03125 to single precision floating point.

19	/2	.	*2	.03125
9	1		0	.0625
4	1		0	.125
2	0		0	.25
1	0		0	.5
0	1		1	0

$10011.00001 = 1.001100001 \times 2^4$

$sign = 0$

$exponent = 127 + 4 = 131 = 10000011_2$

$significant = 00110000100000000000000$ (don't forget the hidden bit)

0	10000011	00110000100000000000000