The University of Alabama in Huntsville
Electrical & Computer Engineering Department
CPE 431 01
Final Exam Solution
Fall 2016

1. (1 point) _**Time**_ is the most important performance metric.
2. (1 point) A _**process**_ includes one or more threads, the address space and the operating system stack.
3. (1 point) _**Synchronization**_ is the process of coordinating the behavior of two or more processes, which may be running on different processors.
4. (1 point) _**Message passing**_ involves communicating between multiple processor by explicitly sending and receiving information.
5. (1 point) MIPS in an example of a _**load/store**_ architecture.
6. (8 points) What number does 0x36E9 A000 represent, assuming the IEE 754 single precision format?

`0x36E9 A000` = 0011 0110 1110 1001 1010 0000 0000 0000
S = 0, number is positive, EXP + BIAS = 0110 1101 = 6 x 16^1 + 13x16^0 = 96 + 13 = 109, EXP = 109 - 127 = -18
Fraction = 0.1101 0011 0100 0000 0000 000
Number is -(1.1101 0011 01)x2^{-18}
= (1110 1001 1010) x 2^{-18} x 2^{-11}) =- 0xE9A x 2^{-29}
= 6.9626 x 10^{-6}

7. (15 points) Here is a series of address references given as hexadecimal word addresses: 21, 4, 8, 5, 20, 37, 19, 5E, 209, 11, 4, 43, 5, 3E, 8, 16, 59, 187, 2E8, 30. Assuming a direct mapped cache with eight word blocks, a total size of 64 words that is initially empty, (a) label each reference in the list as a hit or a miss and (b) show the entire history of the cache, including tag and data.

$$64\ words \times \frac{1\ block}{8\ words} \times \frac{1\ set}{1\ block} = 8\ sets$$

Word Address (Hexadecimal)	Word Address (Decimal)	Binary **Index Block Offset**	Miss/Hit
0x21	33	00 0000 0000 **100 001**	Miss
0x4	4	00 0000 0000 **000 100**	Miss
0x8	8	00 0000 0000 **001 000**	Miss
0x5	5	00 0000 0000 **000 101**	Hit
0x20	32	00 0000 0000 **100 000**	Hit
0x37	55	00 0000 0000 **110 111**	Miss
0x19	25	00 0000 0000 **011 001**	Miss
0x5E	94	00 0000 0001 **011 110**	Miss
0x209	521	00 0000 1000 **001 001**	Miss
0x11	17	00 0000 0000 **010 001**	Miss
0x4	4	00 0000 0000 **000 100**	Hit
0x43	67	00 0000 0001 **000 011**	Miss
0x5	5	00 0000 0000 **000 101**	Miss
0x3E	62	00 0000 0000 **111 110**	Miss
0x8	8	00 0000 0000 **001 000**	Miss
0x16	22	00 0000 0000 **010 110**	Hit
0x59	89	00 0000 0001 **011 001**	Hit
0x187	391	00 0000 0110 **000 111**	Miss
0x2E8	744	00 0000 1011 **101 000**	Miss
0x30	48	00 0000 0000 **110 000**	Hit

Set	**Tag**	**Data**
0	**~~0~~, ~~1~~, ~~0~~, 6**	**~~M[0..7]~~, ~~M[64..71]~~, ~~M[0..7]~~, M[384..391]**
1	**~~0~~, ~~8~~, 0**	**~~M[8..15]~~, ~~M[520..527]~~, M[8..15]**
2	**0**	**M[16..23]**
3	**~~0~~, 1**	**~~M[24..31]~~, M[88..95]**
4	**0**	**M[32..39],**
5	**11**	**M[744..751]**
6	**0**	**M[48..55]**
7	**0**	**M[56..63]**

8.(10 points) Consider the following portions of three programs running at the same time on three processors in

Core 1: `y = 5/z + w;`

Core 2: `x = x + y/(w + 1);`

Core 3: `z = w*(x - y) + z;`

	w	**x**	**y**	**z**
123	**1**	**4**	**2**	**6**
132	**1**	**4**	**2**	**5**
213	**1**	**4**	**2**	**6**
231	**1**	**4**	**1**	**6**
312	**1**	**4**	**2**	**5**
321	**1**	**4**	**2**	**5**

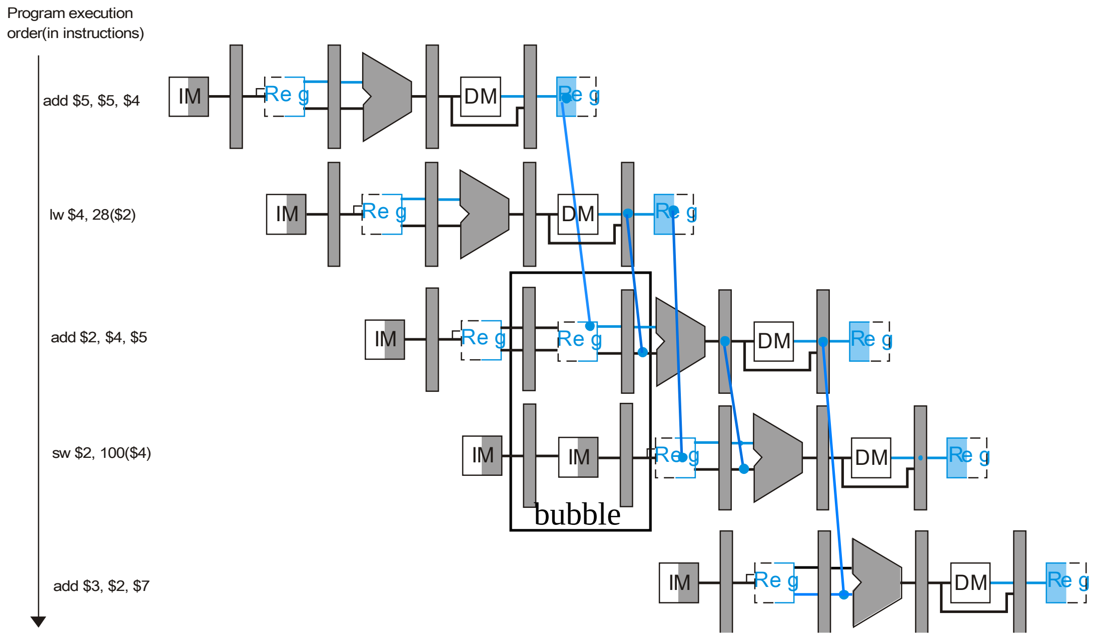
9. (5 points) Assume a program requires the execution of 50 x 10E6 FP operations, 110 x 10E6 INT instructions, 80 X 10E6 L/S instructions and 16 x 10E6 branch instructions. The CPI for each type of instruction is 1, 1, 4, and 2, respectively. Assume that the processor has a 2 GHz clock rate. By how much must we improve the CPI of FP instructions if we want the program to run two times faster?

$ET_{original}$ = (50 x 10E6 * 1 + 110 x 10E6 * 1 + 80 x 10E6 * 4 + 16 x 10E6 * 2)/2 x 10E9
= (512 x 10E6)/2 x 10E9 = 256 ms
$ET_{desired}$ = 128 ms
128 ms = (50 x 10E6 * CPI_{FP_new} + 462 x 10E6)/2 x 10E9
256 x 10E6 = 462 x 10E6 + 50 x 10E6 * CPI_{FP_new}
CPI_{FP_new} = -206 x 10E6/50 x 10E6 = -4.12
∴ It is not possible to make the program run two times faster by only changing the CPI of the FP instructions.

10. (15 points) (a) Identify all of the data dependencies in the following code. (b) How is each data dependency either handled or not handled by forwarding? Draw a multiple clock cycle style diagram to support your answer.

```
a      add   $5, $5, $4
b      lw    $4, 28($2)
c      add   $2, $4, $5
d      sw    $2, 100($4)
e      add   $3, $2, $7
```

```
Dependencies    Hazard?
     a-c        yes       forward from MEM/WB pipeline register before
                          stall from b-c, after stall, data is available
                          in the register file
     b-c        yes       stall, forward from MEM/WB pipeline register
     b-d        no        data is available in register file after stall
     c-d        yes       forward from EX/MEM pipeline register
     c-e        yes       forward from MEM/WB pipeline register
```



11. (10) points Consider an SMT processor that allows instructions from 2 threads to be run concurrently (i.e., there are two functional units), and instructions from either or both threads can be issued to run on any cycle. Assume we have two threads X and Y to run on these CPUs that include the following operations:

Thread X	Thread Y
A1 – takes 3 cycles to execute A2 – no dependences A3 – can't be executed at the same time as A1 A4 – depends on the result of A3 A5 – no dependences and takes two cycles to execute	B1 – take 2 cycles to execute B2 – can't be executed at the same time as B1 B3 – depends on the result of B2 B4 – no dependences and takes 2 cycles to execute B5 – depends on the results of B1 and takes 3 cycles to execute

Assume all operations take a single cycle to execute unless noted otherwise or they encounter a hazard. How many cycles will it take to execute these two threads? How many issue slots are wasted due to hazards?

Issue Slot	**FN1**	**FN2**
1	**A1**	**B1**
2	**A1**	**B1**
3	**A1**	**B2**
4	**A2**	**B3**
5	**A3**	**B4**
6	**A4**	**B4**
7	**A5**	**B5**
8	**A5**	**B5**
9		**B5**

12. (10 points For the MIPS assembly instructions below, what is the corresponding C statement? 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   $t0, $s0, 2          // $t0 = f*4
add   $t0, $s6, $t0        // $t0 = &A[0] + f*4 (&A[f])
sll   $t1, $s1, 2          // $t1 = g*4
add   $t1, $s7, $t1        // $t1 = &B[0] + g*4 (&B[g])
lw    $t2, 0($t0)          // $t2 = A[f]
addi  $t0, $t0, 4          // $t0 = &A[f+1]
lw    $t0, 0($t0)          // $t0 = A[f+1]
add   $t0, $t0, $t2        // $t0 = A[f+1] + A[f]
sw    $t0, 0($t1)          // B[g] = A[f+1] + A[f]
```

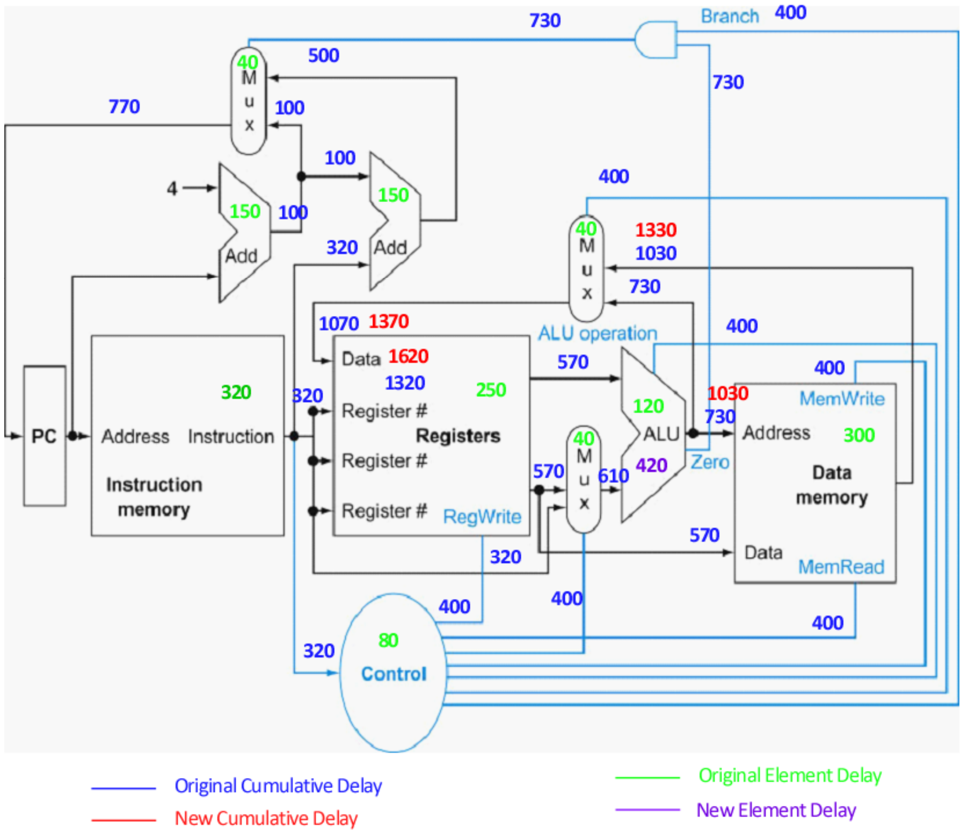
```
B[g] = A[f+1] + A[f];
```

13. (7 points) Given a virtual memory system with 48 bit virtual addresses, 16 KiB pages, and 8 byte page table entries. Calculate the total page table size for a system running 7 applications that utilize a third of the memory available.

Page offset = $\log_2$ 16 KiB = 14 bits, The number of virtual page numbers is $2^{48-14} = 2^{34}$
Page table size = Number of virtual pages x page table entry size x number of applications x utilization
Page table size = 2^{34} x 8 bytes x 7 x 1/3 = 16 GiB x 8 x 7 x 1/3 = 298.67 GiB

14. (15 points)When processor designers consider a possible improvement to the processor datapath, the decision usually depends on the cost/performance trade-off. In the following three problems, assume that we are starting with the datapath shown, where I Mem, Add, Mux, ALU, Regs, D Mem, and Control blocks have latencies of 320 ps, 150 ps, 40 ps, 120 ps, 250 ps, 300 ps, and 80 ps, respectively, and costs of 1000, 30, 10, 100, 200, 2000, and 500, respectively.

Consider the addition of a multiplier to the ALU. This addition will add 300 ps to the latency of the ALU and will add a cost of 600 to the ALU. The result will be 5% fewer instructions executed since we will no longer need to emulate the MUL instruction.



What is the speedup achieved by adding this improvement?

$$\frac{P_{new}}{P_{old}} = \frac{ET_{old}}{ET_{new}}$$

$$= \frac{IC_{old} * CPI_{old} * CT_{old}}{IC_{new} * CPI_{old} * CT_{new}} = \frac{IC_{old} * 1 * 1320\,ps}{0.95 IC_{old} * 1 * 1620\,ps} = 0.86$$