The University of Alabama in Huntsville
Electrical & Computer Engineering Department
CPE 431 01
Final Exam Solution
Fall 2015

1. (1 point) _**Fine-grained**_ multithreading is a version of hardware multithreading that implies switching between threads after every instruction.
2. (1 point) A _**thread**_ is a lightweight process, groups of them commonly share a single address space.
3. (1 point) A _**lock**_ is a synchronization device that allows only one processor to access data at a time.
4. (1 point) _**Open MP**_ is an API for shared memory multiprocessing that includes compiler directives, a library and runtime directives.
5. (1 point) _**GPUs**_ usually don't have caches.
6. (7 points) What number does 0x3FE9 A700 0000 0000 0000 0000 0000 0000 represent, assuming the IEE 754 double precision format?

0011 1111 1110 1001 1010 0111 0000 0000 …
S = 0, number is positive
Exponent = 011 1111 1110 = 1022
Fraction = 1001 1010 0111 0000 0000 0000 0000 0000 0000 0000 0000 0000
Bias = 1023

Number = $(-1)^S \times (1 + \text{Fraction}) \ 2^{\text{Exponent} - \text{Bias}}$
$= (-1)^0 \times (2^{52} + 2^{51} + 2^{48} + 2^{47} + 2^{45} + 2^{42} + 2^{41} + 2^{40})/2^{52} \times 2^{1022-1023}$
$= (2^{52} + 2^{51} + 2^{48} + 2^{47} + 2^{45} + 2^{42} + 2^{41} + 2^{40})/2^{52} \times 2^{-1}$
$= (2^{52} + 2^{51} + 2^{48} + 2^{47} + 2^{45} + 2^{42} + 2^{41} + 2^{40})/2^{53}$
$= 0.8016$

7. (15 points) Here is a series of address references given as hexadecimal word addresses: 1, 4, 8, 5, 20, 17, 19, 56, 209, 11, 4, 43, 5, 36, 8, 16, 59, 187. Assuming a direct mapped cache with eight word blocks, a total size of 32 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.

Word Address (Hexadecimal)	Word Address (Decimal)	Binary, Tag = bits (31..5) -only bits(11..5) shown, Index = bits(4..3), Block Offset = bits(2..0)	Miss/Hit
0x1	1	**0000 000 00 001**	**miss**
0x4	4	**0000 000 00 100**	**hit**
0x8	8	**0000 000 01 000**	**miss**
0x5	5	**0000 000 00 101**	**hit**
0x20	32	**0000 001 00 000**	**miss**
0x17	23	**0000 000 10 111**	**miss**
0x19	25	**0000 000 11 001**	**miss**
0x56	86	**0000 010 10 110**	**miss**
0x209	521	**0010 000 01 001**	**miss**
0x11	17	**0000 000 10 001**	**miss**
0x4	4	**0000 000 00 100**	**miss**
0x43	67	**0000 010 00 011**	**miss**
0x5	5	**0000 000 00 101**	**miss**
0x36	54	**0000 001 10 110**	**miss**
0x8	8	**0000 000 01 000**	**miss**
0x16	22	**0000 000 10 110**	**miss**
0x59	89	**0000 010 11 001**	**miss**
0x187	295	**0001 100 00 111**	**miss**

$$32words \times \frac{1block}{8words} \times \frac{1set}{1blocks} = 4sets$$

3 bits block offset, 2 bits index

Index	Tag (Decimal)	Data (Decimal)
0	**~~0~~, ~~1~~, ~~0~~, ~~2~~, ~~0~~, 12**	**~~M[0..7],~~ ~~M[32..39],~~ ~~M[0..7]~~, ~~M[64..71],~~ ~~M[0..7],~~ M[288.295]**
1	**~~0~~, ~~16~~, 0**	**~~M[8..15],~~ ~~M[520..527],~~ M[8..15],**
2	**~~0~~, ~~2~~, ~~0~~, ~~1~~, 0**	**~~M[16..23],~~ ~~M[80..87],~~ ~~M[16..23],~~ ~~M[48..55],~~ M[16..23]**
3	**~~0~~, 2**	**~~M[24..31],~~ M[88..95]**

8. (10 points) Consider the following portions of three programs running at the same time on three processors in a symmetric multicore processor (SMP). Assume that before this code is run, w is 2, x is 4 and y is 3 and z is 1. w, x, y, and z are type `int`. What are all of the possible outcomes of executing these instructions?

Core 1: `y = 5/z + w;`
Core 2: `x = x + y/(w + 1);`
Core 3: `z = w*(x - y) + z;`

Order	w	x	Y	z
1, 2, 3	2	6	7	-1
1, 3, 2	2	6	7	-5
2, 1, 3	2	5	7	-3
2, 3, 1	2	5	3	5
3, 1, 2	2	5	3	3
3, 2, 1	2	5	3	3

9. (5 points) Sometimes software optimization can dramatically improve the performance of a computer system. Assume that a CPU can perform a multiplication operation in 10 ns, and a subtraction operation in 1 ns. (a) How long will it take for the CPU to calculate the result of d = a × b – a × c? (b) Could you optimize the equation so that it will take less time? (c) If so, what is the resulting speedup?

a. 2 multiplications + 1 subtraction = 20 ns + 1 ns = 21 ns
b. d = a x b – a x c = a(b-c), 1 subtraction + 1 multiplication = 1 ns + 10 ns = 11 ns
c. 21 ns/11 ns = 1.9

10. (10 points) In this exercise we look at memory locality properties of matrix computation. The following code is written in MATLAB, where elements within the same column are stored contiguously. Assume each word is a 32-bit integer.

```
for I = 1:8
  for J = 1:8000
    A(I, J) = B(I, 1) + A(J, J);
```

For each variable, indicate whether references to them exhibit spatial or temporal locality. Explain your answers.

Variable	I	J	A(I ,J)	B(I, 1)	A(J, J)
Spatial	Unknown	Unknown	No, elements are accessed by row	Yes, elements are accessed by column	No, each access comes from a different column
Temporal	Yes, used 64000 times to access array elements	Yes, used 64000 times to access array elements	No, a different element is accessed each time, unless once every 8000 times counts	Yes, used 8000 times, each	No, a different element is accessed each time, unless once every 8000 time counts

11. (10 points) Consider a SEC code that protects 8 bit words with 4 parity bits. If we read the value 0x374, is there an error? If so, correct the error.

```
PPDP DDDP DDDD
1234 5678 9111
          012
0011 0111 0100
```

C1 = P1 ⊕ D3 ⊕ D5 ⊕ D7 ⊕ D9 ⊕ D11 = 0 ⊕ 1 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 0 = 0
C2 = P2 ⊕ D3 ⊕ D6 ⊕ D7 ⊕ D10 ⊕ D11 = 0 ⊕ 1 ⊕ 1 ⊕ 1 ⊕ 1 ⊕ 0 = 0
C4 = P4 ⊕ D5 ⊕ D6 ⊕ D7 ⊕ D12 = 1 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 0 = 1
C8 = P8 ⊕ D9 ⊕ D10 ⊕ D11 ⊕ D12 = 1 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 0 = 0
There is one incorrect bit, bit 4, corrected is 0x274.

12. (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 – conflicts for a functional unit with A1 A4 – depends on the result of A3	B1 – take 2 cycles to execute B2 – conflicts for a functional unit with B1 B3 – depends on the result of B2 B4 – no dependences and takes 2 cycles to execute

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

Cycle	Functional Unit 0	Functional Unit 1
1	A1	B1
2	A1	B1
3	A1	A2
4	B2	A3
5	A4	B4
6	B3	B4

It will take 6 cycles, no issue slots are wasted.

13. (10 points) Consider the following MIPS loop:

```
        add   $t2, $zero, $zero
        addi  $s1, $zero, 1
        addi  $t1, $zero, 5
LOOP:   slt   $t4, $zero, $t1
        beq   $t4, $zero, DONE
        addi  $t1, $t1, -1
        addi  $s1, $s1, 2
        mul   $t3, $s1, $s1
        add   $t2, $t2, $t3
        j     LOOP
DONE:
```

What is the value in register $t2 upon completion of the loop execution? Write the equivalent C. Assume that registers `$s1`, `$s2`, `$t1`, and `$t2` are integers `A`, `B`, `i` and `temp`, respectively.

```
temp = 0;
A = 1;
for (i = 5; i > 0; i--)
{
  A = A + 2;
  temp = temp + A * A;
}
```

First iteration: temp = 0 + 3*3 = 9
Second iteration: temp = 9 + 5*5 = 34
Third iteration: temp = 34 + 7*7 = 83
Fourth iteration: temp = 83 + 9*9 = 164
Fifth iteration: temp = 164 + 11*11 = 285 (value of $t2 upon loop completion)

14. (18 points) Consider a pipeline for a register-memory architecture. The architecture has two instruction formats: a register-register format and a register-memory format. There is a single memory addressing mode (offset + base register). There is a set of ALU operations as follows:

ALUop← Rdest, Rsrc1, Rsrc2, offset (Rdest ← Rsrc1 ALUop Rsrc2 or
Rdest ← Rsrc1 ALUop MEM[Rsrc2 + offset])
Rdest ← MEM[Rsrc2 + offset]
MEM[Rsrc2 + offset] ← Rsrc1
where the ALUop is one of the following and Rdest, Rsrc1 and Rsrc2 are registers:

- Add, Subtract, And, Or(with or without offset)
- Load (Rsrc1 omitted)
- Store (Rdest omitted)

Branches use a full compare of two registers and are PC-relative. Assume that this machine is pipelined so that a new instruction is started every clock cycle. The pipeline structure is

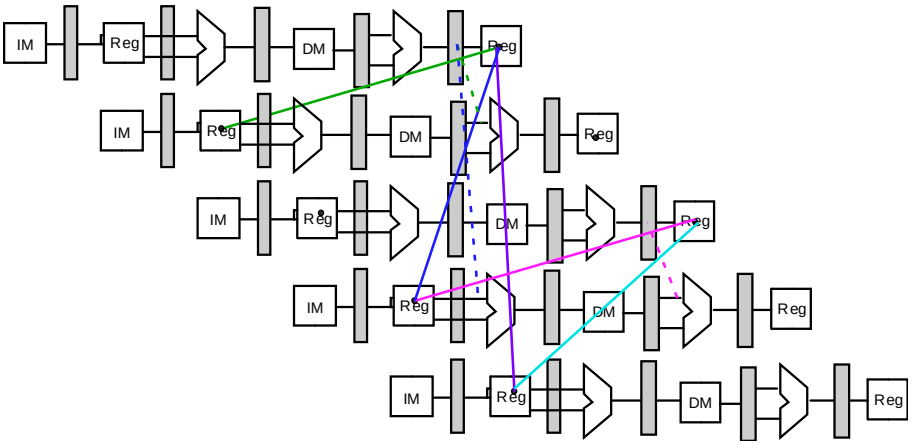
IF	RF	ALU1	MEM	ALU2	WB					
	IF	RF	ALU1	MEM	ALU2	WB				
		IF	RF	ALU1	MEM	ALU2	WB			
			IF	RF	ALU1	MEM	ALU2	WB		
				IF	RF	ALU1	MEM	ALU2	WB	
					IF	RF	ALU1	MEM	ALU2	WB

The first ALU stage is used for effective address calculation for memory references and branches. The second ALU stage is used for operations and branch comparisons. RF is both a decode and register-fetch stage. Assume that when a register read and a register write of the same register occur in the same clock cycle, the write data is forwarded. (a) (10 points) For the following code fragment:

```
(a)    add      $1, $1, 0($15)
(b)    add      $3, $1, 4($15)
(c)    add      $2 ,$2, 0($12)
(d)    add      $4, $2, 4($1)
(e)    store    $1, 8($2)
```

identify the data dependencies and draw a figure (using the multiple clock style) showing them as lines. Assume that there are separate ALUs for the ALU1 and ALU2 pipe stages.
(b) (8 points) Which of these dependencies could be solved by forwarding? Identify the dependencies and describe the forwarding.

(a) **Dependences - 1: (a)-(b), 2: (a)-(d), 3: (a)-(e), 4: (c)-(d), 5: (c)-(e)**



(b) 1 and 4 could be solved by forwarding from ALU2/WB to ALU2. (see dotted lines)
3 is not a hazard
2 can be solved by forwarding from ALU2/WB to ALU1 (see dotted line)
5 can be solved by forwarding from ALU2/WB to ALU1 only after stalling