The University of Alabama in Huntsville
Electrical & Computer Engineering Department
CPE 431 01
Final Exam solution
Fall 2018

1. (1 point) _**Task/process**_ level parallelism utilizes multiple processors by running independent programs simultaneously.
2. (1 point) _**Coarse-grained**_ multi-threading is a version of hardware multithreading that implies switching between threads only after significant events.
3. (1 point) A _**lock**_ is a synchronization device that allows only one processor to access data at a time.
4. (2 points) The two common data sharing mechanisms for multiprocessors are _shared memory_ and _**message passing**_.

5. (8 points) What number does 0xC39B A000 0000 0000 represent, assuming the IEEE 754 double precision format? Express the answer in decimal.

```
0xC39B A000 0000 0000 = 1100 0011 1001 1011 1010 1000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
1 100 0011 1001 1011 1010 1000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000
```

S = 1, the number is negative

Stored exponent is 4 * 256 + 3* 16 + 9 = 1081, subtracting 1023 gives 58. The fraction is 1011 1010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 $= (2^{51} + 2^{49} + 2^{48} + 2^{47} + 2^{45})/2^{52} = 0.7265625$

Number is -1.7265625 × 2^{58} = -4.976 E17

6. (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, 16, 59, 42, 30. Assuming a direct mapped cache with two word blocks, a total size of 16 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.
16 words x 1 block/2 words x 1 set/1 block = 8 sets, 3 bits index, 1 bit block offset

Word Address (Hexadecimal)	Word Address (Decimal)	Binary	Miss/Hit
0x21	33	0000 0000 0010 **000** 1	**Miss**
0x4	4	0000 0000 0000 **010** 0	**Miss**
0x8	8	0000 0000 0000 **100** 0	**Miss**
0x5	5	0000 0000 0000 **010** 1	**Hit**
0x20	32	0000 0000 0010 **000** 0	**Hit**
0x37	55	0000 0000 0011 **011** 1	**Miss**
0x19	25	0000 0000 0001 **100** 1	**Miss**
0x5E	94	0000 0000 0101 **111** 0	**Miss**
0x209	521	0000 0010 0000 **100** 1	**Miss**
0x11	17	0000 0000 0001 **000** 1	**Miss**
0x4	4	0000 0000 0000 **010** 0	**Hit**
0x43	67	0000 0000 0100 **001** 1	**Miss**
0x5	5	0000 0000 0000 **010** 1	**Hit**
0x3E	62	0000 0000 0011 **111** 0	**Miss**
0x16	22	0000 0000 0001 **011** 0	**Miss**
0x59	89	0000 0000 0101 **100** 1	**Miss**
0x42	66	0000 0000 0100 **001** 0	**Hit**
0x30	48	0000 0000 0011 **000** 0	**Miss**

Index	**Tag**	**Data**
0	**~~2~~, ~~1~~, 3**	**~~M[32..33]~~, ~~M[16..17]~~, M[48..49]**
1	**4**	**M[66..67]**
2	**0**	**M[4..5]**
3	**~~3~~, 1**	**~~M[54..55]~~, M[22..23]**
4	**~~0, 1, 32,~~ 5**	**~~M[8..9], M[24..25], M[520..521].~~ M[88..89]**
5		
6		
7	**~~5,~~ 3**	**~~M[94..95],~~ M[62..63]**

7. (9 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, the `int` variables w, x, y, z, are 4, 2, 2, 3, respectively. What are all the possible outcomes of executing these instructions?

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

	w	x	y	z
123	4	16	6	43
132	4	16	6	-13
213	4	8	6	11
231	4	8	5	27
312	4	16	6	3
321	4	8	6	3

8. (5 points) Assume that registers `$s0` and `$s1` hold the values `0xFFFF_FFFF` and `0xFFFF_F800`, respectively and that these values represent signed integers.

a. (3 points) What is the value of `$t0` for the following assembly code?

```
add   $t0, $s0, $s1
```

b. (2 points) Is the result in `$t0` the desired result, or has there been overflow?

```
 $s0 1111 1111 1111 1111 1111 1111 1111 1111
+$s1 1111 1111 1111 1111 1000 0000 0000 0000
     1111 1111 1111 1111 0111 1111 1111 1111 S
   1 1111 1111 1111 1111 0000 0000 0000 0000 C
```

No overflow

9. (5 points) If the current value of the PC in a MIPS processor is `0x0000_0600`, can you use a single branch instruction to get to the PC address `0x2001_4924`? Explain your answer.

Branch Target Desired = 0x2001 4924

For a current PC of 0x0000 0600, we can branch to 0x0000 0604 + 0x0001 FFFC = 0x0002 0600 and 0x0000 0604 + 0xFFFE 0000, or 0xFFFE 0604. That represents the memory 0x0000 0000 – 0x0002 0600 and 0XFFFE 0604 – 0xFFFF FFFC. So, no, the range doesn't include 0x2001 4924.

10. (8 points) Multilevel caching is an important technique to overcome the limited amount of space that a first level cache can provide while still maintaining its speed. Consider a processor with the following parameters.

Base CPI, no memory stalls	Processor speed	Main memory access time	First-level cache miss rate per instruction	Second-level cache, direct-mapped speed	Global miss rate with second-level cache, direct-mapped	Second-level cache, eight-way s et associative speed	Global miss rate with second-level cache, eight-way set associative
1.5	**2 GHz**	**100 ns**	**7 %**	**12 cycles**	**3.5 %**	**28 cycles**	**1.5 %**

A designer wants to use the second level eight-way set associative cache and add a third level cache. The third level cache takes 40 cycles to access and will reduce the global miss rate to 0.9%. What is the CPI for the total system with the addition of this third level cache?
CPI total = CPI_{base} +$L1_{hit}$ + $L1_{miss}$*$L2_{hit}$ + $L2_{miss}$*$L3_{hit}$ + $L3_{miss}$*main memory access = 1.5 + 0 + 0.07*28 + 0.015*40 + 0.009*100ns*2E9 cycles/s = 1.5 + 1.96 + 0.6 + 1.8 = 5.86

11. (6 points) The following list provides parameters of a virtual memory system.

Virtual Address (bits)	Physical DRAM Installed	Page Size	PTE Size (byte)
52	64 GiB	8 KiB	8

Using a multilevel page table can reduce the physical memory consumption of page tables, by only keeping active PTEs in physical memory.

(a) (5 points) How many levels of page tables will be needed in this case?
#Virtual pages = $2^{52}/2^{13} = 2^{39}$, Each page contains $2^{13}/2^{3} = 2^{10}$ page table entries
$\lceil 39/10 \rceil$ = 4 levels of page tables

(b) (1 point) how many memory references are needed for address translation if missing in TLB? **4, same as the number of levels**

12. (4 points) Assume that individual stages of a MIPS datapath have the following latencies:

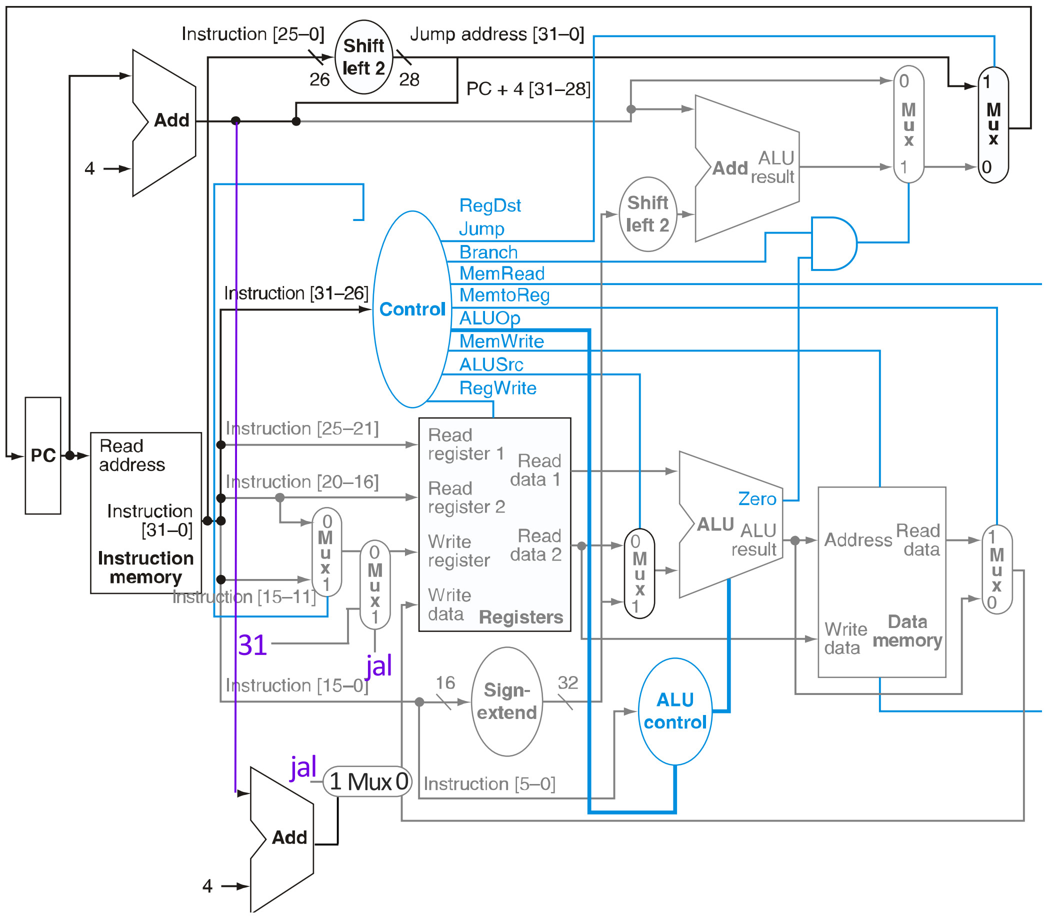
IF	ID	EX	MEM	WB
250 ps	350 ps	150 ps	400 ps	200 ps

a. (3 points) What is the clock cycle time in a pipelined and non-pipelined processor?
Pipelined = max (IF, ID, EX, MEM, WB) = max (250, 350, 150, 400, 200) ps = 400 ps
Non-Pipelined = sum (IF, ID, EX, MEM, WB) = (250 + 350 + 150 + 400 + 200) ps = 1350 ps

b. (1 point) If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?
Split the longest stage (MEM) into MEM0, MEM1
Pipelined = max (IF, ID, EX, MEM0, MEM1, WB) = max(250, 350, 150, 200, 200, 200) ps = 350 ps

13. (10 points) Add the instruction `jal` (jump and link) to the single-cycle datapath shown in the figure below. The `jal` instruction is a J-type instruction and its operation is defined below. Add any necessary datapaths and control signals and show the necessary additions to the table of control signals given.

`jal jump_address` PC ← jump_address
$ra ← PC + 8



Instruction	RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0	jump	jal
R-format	1	0	0	1	0	0	0	1	0	0	0
lw	0	1	1	1	1	0	0	0	0	0	0
sw	d	1	d	0	0	1	0	0	0	0	0
beq	d	0	d	0	0	0	1	0	1	0	0
jal	d	d	d	1	0	0	0	d	d	1	1

14. (10 points) Consider adding an addressing mode to MIPS that allows arithmetic instructions to directly access memory, as is found on the 80x86. The primary benefit is that fewer instructions will be executed because we won't have to first load a register. The primary disadvantage is that the cycle time will have to increase to account for the additional time to read memory. Consider adding a new instruction:

```
addm $t2, 100($t3) # $t2 = $t2 + Memory[$t3 + 100]
```

Assume that the new instruction will cause the cycle time to increase by 15 %. Use the instruction frequencies given, and assume that three-fifths of the data transfers are loads and the rest are stores. Assume that the new instruction affects only the clock speed, not the CPI. What percentage of loads must be eliminated for the machine with the new instruction to have at least the same performance?

Instruction Class	Frequency
Arithmetic	48 %
Data Transfer	33 %
Conditional branch	17 %
Jump	2 %

***x is the % of loads affected**

CC_{mod} = 1.15 $CC_{orig,}$ CPI_{mod} = CPI_{orig}=CPI, IC_{mod} = IC_{orig}(1- 0.33*0.6*x)

ET = IC*CPI*CC. ET_{orig} = IC_{orig}* CPI_{orig}*CC_{orig}, ET_{mod} = IC_{mod}* CPI_{mod}*CC_{mod}

For break even point,

ET_{orig} = ET_{mod}

IC_{orig} * CPI * CC_{orig} = IC_{mod} * CPI * 1.15 * CC_{orig}

IC_{orig} = IC_{mod} * 1.15

Ic_{orig} = Ic_{orig}(1-0.198x) * 1.15

$$\frac{1}{1.15} = 1 - 0.198x$$

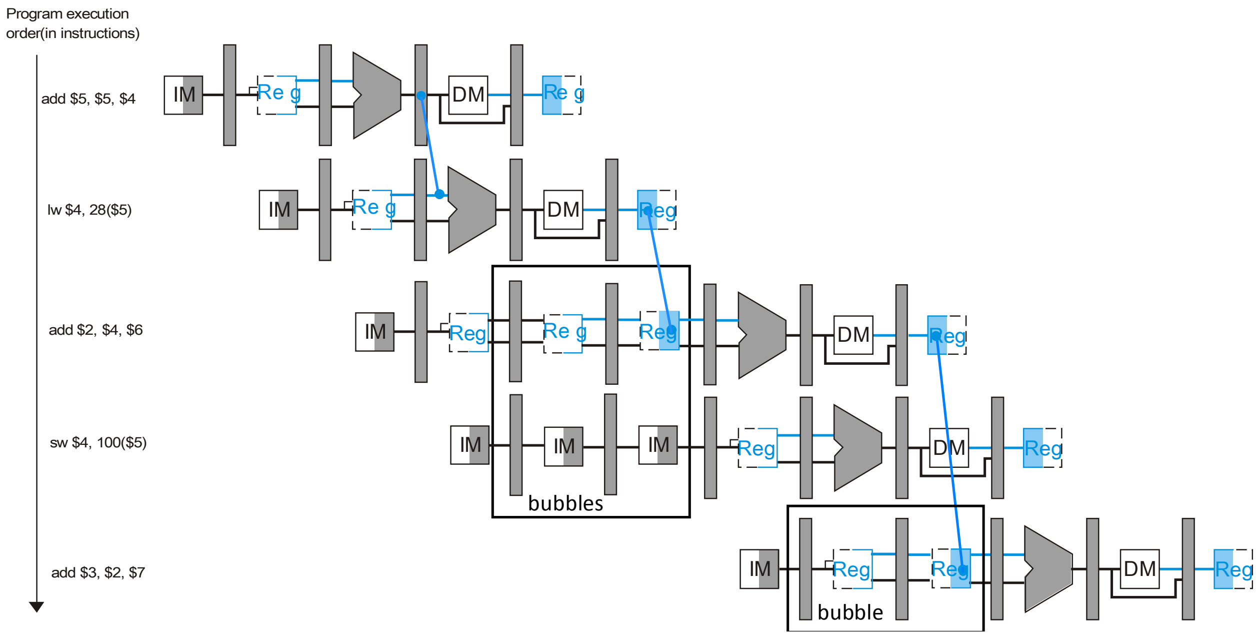
0.198x = 1- $\frac{1}{1.15}$

x = 65.9

15. (15 points) (a) (5 points) Identify all of the data dependencies in the following code. (b) (10 points) How is each data dependency either handled or not handled by **forwarding from the EX/MEM pipeline register only**? Draw a multiple clock cycle style diagram to support your answer.

```
a      add   $5, $5, $4
b      lw    $4, 28($5)
c      add   $2, $4, $6
d      sw    $4, 100($5)
e      add   $3, $2, $7
```

Dependencies	Handled by EX/MEM forwarding?	
a-b	yes	
a-d	no	previous stalls take care of it
b-c	no	data is available in register file after 2 stalls
b-d	no	b-c stall takes care of it
c-e	no	data is available in register file after 1 stall


Program execution order(in instructions)
add $5, $5, $4
lw $4, 28($5)
add $2, $4, $6
sw $4, 100($5)
add $3, $2, $7
IM
Reg
DM
bubbles
bubble