The University of Alabama in Huntsville
Electrical & Computer Engineering Department
CPE 431 01
Final Exam Solution
Fall 2019

You must show your work to receive full credit!!

1. (1 point) True/False _**True**_ Power efficiency is a motivation for using multiprocessors.
2. (1 point) True/False _**True**_ Memory access patterns of various programs can have a dramatic impact on the performance of those programs.
3. (1 point) The primary difficulty in exploiting the full potential of parallelism is in the _**software**_
4. (1 point) _**Fine-grained**_ multithreading switches between threads on each instruction, often done round robin.
5. (1 point) OpenMP extends C using _**pragmas**_, commands to the C macro processor.
6. (8 points) What number does 0x79E2 C000 0000 0000 represent, assuming the IEEE 754 double precision format? Express the answer in decimal.

   0x79E2 C000 0000 0000 = 0111 1001 1110 0010 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

   0 111 1001 1110 0010 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

   S = 0, the number is positive

   Stored exponent is 7 * 256 + 9* 16 + 14 = 1950, subtracting 1023 gives 927. The fraction is 0010 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 = $(2^{49} + 2^{47} + 2^{46})/2^{52}$ = 0.171875

   Number is 1.171875 × 2^{927} = 1.329 E279

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, 22, 18, 16, 59, 42, 30. Assuming a direct mapped cache with four 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.

32 words x 1 block/4 words x 1 set/1 block = 8 sets, 3 bits index, 2 bits block offset

Word Address (Hexadecimal)	Word Address (Decimal)	Binary	Miss/Hit
0x21	33	0000 0000 001 **000** 01	**Miss**
0x4	4	0000 0000 000 **001** 00	**Miss**
0x8	8	0000 0000 000 **010** 00	**Miss**
0x5	5	0000 0000 000 **001** 01	**Hit**
0x20	32	0000 0000 001 **000** 00	**Hit**
0x37	55	0000 0000 001 **101** 11	**Miss**
0x19	25	0000 0000 000 **110** 01	**Miss**
0x5E	94	0000 0000 010 **111** 10	**Miss**
0x209	521	0000 0010 000 **010** 01	**Miss**
0x11	17	0000 0000 000 **100** 01	**Miss**
0x7	7	0000 0000 000 **001** 11	**Hit**
0x43	67	0000 0000 010 **000** 11	**Miss**
0x5	5	0000 0000 000 **001** 01	**Hit**
0x22	34	0000 0000 001 **000** 10	**Miss**
0x18	24	0000 0000 000 **110** 00	**Hit**
0x16	22	0000 0000 000 **101** 10	**Miss**
0x59	89	0000 0000 010 **110** 01	**Miss**
0x42	66	0000 0000 010 **000** 10	**Miss**
0x30	48	0000 0000 001 **100** 00	**Miss**

Index	**Tag* last hex 3 bits**	**Data**
0	~~0x001~~, ~~0x002~~, ~~0x001~~, 0x002	~~M[32:35]~~, ~~M[64:67]~~, ~~M[32:35],~~ M[64:67]
1	0x000	M[4:7]
2	~~0x000~~, 0x020	~~M[8:11],~~ M[520:523]
3		
4	~~0x000~~, 0x001	~~M[16:19]~~, M[48:51]
5	~~0x001~~, 0x000	~~M[52:55]~~, M[20:23]
6	~~0x000~~, 0x002	~~M[24:27]~~, M[88:91]
7	0x002	M[92:95]

8. (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 2, 5, 3, 2, respectively. What are all the possible outcomes of executing these instructions?

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

Core 2: `x = x + yw;`

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

	w	**x**	**y**	**z**
123	**2**	**11**	**3**	**18**
132	**2**	**11**	**3**	**6**
213	**2**	**11**	**3**	**18**
231	**2**	**11**	**0**	**18**
312	**2**	**7**	**1**	**6**
321	**2**	**11**	**1**	**6**

9. (5 points) Assume that registers `$s0` and `$s1` hold the values `0x7999 6947` and `0x8000 AB59`, respectively and that these values represent signed integers.

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

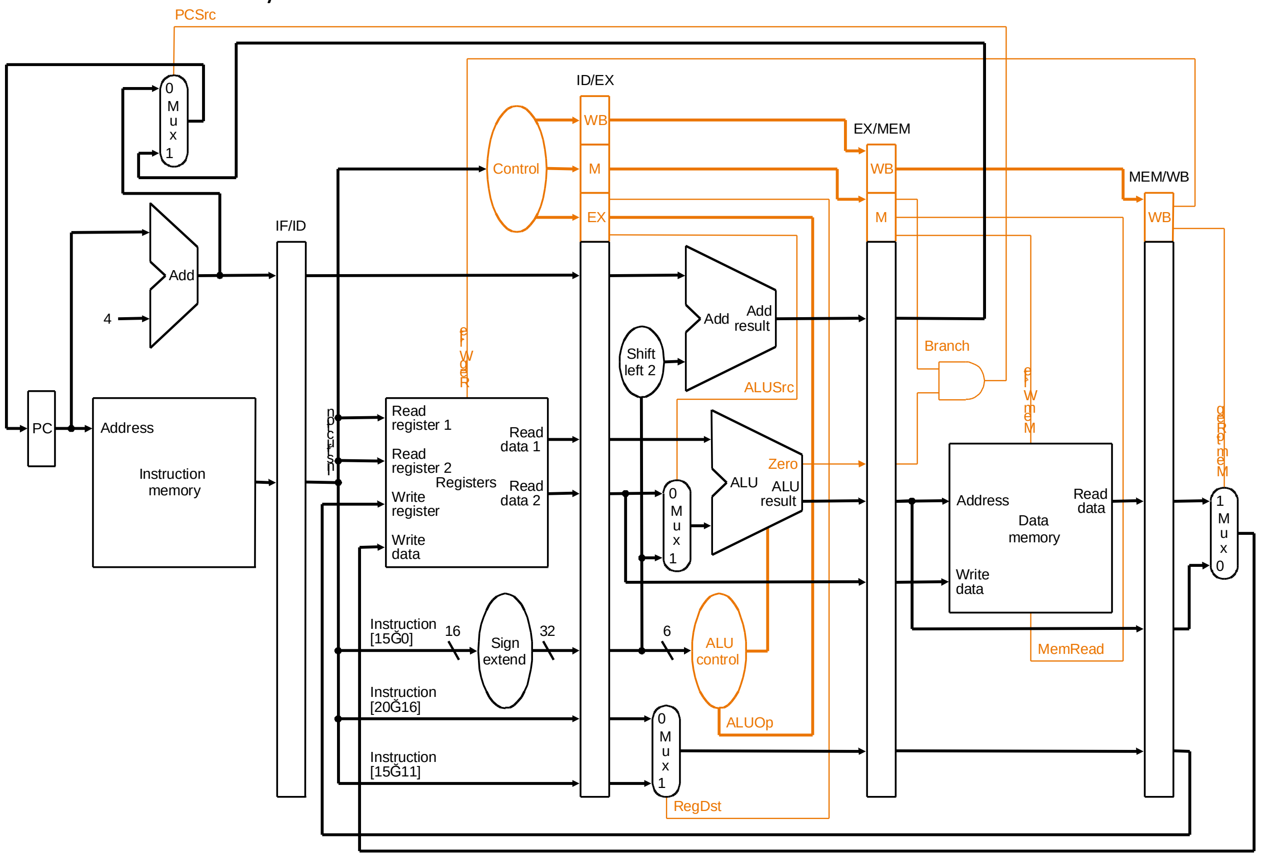
```
add   $t0, $s0, $s1
```

```
$s0                0111 1001 1001 1001 0110 1001 0100 0111
$s1                1000 0000 0000 0000 1010 1011 0101 1001
$t0 = $s0 + $s1    1111 1001 1001 1010 0001 0100 1010 0000 =
  0XF99A 14A0
```

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

No overflow, carry into the sign bit is 0, carry bit out of the sign bit is 0, 0 ⊕ 0 = 0.

10. (15 points) Consider executing the following code on a pipelined datapath like the one shown except that 1) it supports j instructions that complete in the ID stage, and 2) it has MEM/WB forwarding only. The register file supports writing in the first half cycle and reading in the second half cycle.



```
sort:     addi  $sp, $sp, -20
          sw    $ra, 16($sp)
          sw    $s3, 12($sp)
          sw    $s2, 8($sp)
          sw    $s1, 4($sp)
          sw    $s0, 0($sp)
          add   $s2, $a0, $zero
          add   $s3, $a1, $zero
          add   $s0, $zero, $zero
for1tst:  slt   $t0, $s0, $s3
          beq   $t0, $zero, exit1
⇒         addi  $s1, $s0, -1
for2tst:  slt   $t0, $s1, $zero
          bne   $t0, $zero, exit2
          add   $t1, $s1, $s1
          add   $t1, $t1, $t1
200       add   $t2, $s2, $t1
204       lw    $t3, 0($t2)
```

```
208          lw    $t4, 4($t2)
212          slt   $t0, $t4, $t3
216          beq   $t0, $zero, exit2
220          add   $a0, $s2, $zero
224          add   $a1, $s1, $zero
228          jal   swap
232          addi  $s1, $s1, -1
236          j     for2tst
240 exit2:   addi  $s0, $s0, 1
244          j     for1tst
248 exit1:   lw    $s0, 0($sp)
252          lw    $s1, 4($sp)
256          lw    $s2, 8($sp)
260          lw    $s3, 12($sp)
264          lw    $ra, 16($sp)
268          addi  $sp, $sp, 20
272          jr    $ra
```

If the `addi $s1` instruction one instruction before the `for2tst` label begins executing in cycle 1 and the `bne $t0, $zero, exit2` instruction is taken, what instructions are found in each of the five stages of the pipeline in the 12th cycle? Show the instructions being executed in each stage of the pipeline during each cycle. What value is stored in the Instruction field of the IF/ID pipeline register in the 12th cycle? Assume that before the instructions are executed, the state of the machine was as follows:
The PC has the value 200_{10}, the address of the `add $t2` instruction
Every register has the initial value 20_{10} plus the register number.
Every memory word accessed as data has the initial value 10000_{10} plus the byte address of the word.

Cycle	IF	ID	EX	MEM	WB
1	addi $s1				
2	slt $t0	addi $s1			
3	bne $t0	slt $t0	addi $s1		
4	bne $t0	slt $t0	bubble	addi $s1	
5	add $t1	bne $t0	slt $t0	bubble	addi $s1
6	add $t1	bne $t0	bubble	slt $t0	bubble
7	add $t1	add $t1	bne $t0	bubble	slt $t0
8	add $t2	add $t1	add $t1	bne $t0	bubble
9	addi $s0	bubble	bubble	bubble	bne $t0
10	j for1tst	addi $s0	bubble	bubble	bubble
11	lw $s0	j for1tst	addi $s0	bubble	bubble
12	slt $t0	bubble	j for1tst	addi $s0	bubble

IF/ID.Instruction = `lw $s0, 0($sp)   0X8FB0 0000`

Opcode = 35, rs = 29, rt = 16, offset = 100011 11101 10000 0000000000000000
1000 1111 1011 0000 0000 0000 0000 0000 0000 = 0x8FB0 0000

11. (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
1.2	2 GHz	80 ns	6 %	15 cycles	2.5 %	22 cycles	1.5 %

A designer wants to use the second level direct-mapped cache and add a third level cache. The third level cache takes 36 cycles to access and will reduce the global miss rate to 1 %. What is the CPI for the total system with the addition of this third level cache?

$\text{CPI} = \text{CPI}_{\text{base}} + \text{L1}_{\text{miss}}*\text{L1}_{\text{penalty}} + \text{L2}_{\text{miss}}*\text{L2}_{\text{penalty}} + \text{L3}_{\text{miss}}*\text{Memory access}$
$= 1.2 + 0.06*15 + 0.025*36 + 0.01*(80\text{ ns}*2\text{ E9 cycles/s}) = 4.6$

12. (4 points) Write a minimal sequence of actual MIPS instructions to accomplish the same thing as the following pseudoinstruction: `mov $t5, $s0` ($t5 $\leftarrow$ $s0)
`add $t5, $zero, $zero`

13. (5 points) If the current value of the PC is 0x0000 0000, can you use a single jump instruction to get to the 0x0FFF FFA8. If so, specify the fields of the J-type instruction.
PC+4(31..28) = 0000
The maximum value of the jump field is 26 1s which are then shifted left 2, the minimum is 26 0s which are then shifted left 2. The range is 0x0000 0000 to 0x0FFF FFFC -0x0FFF FFA8 falls in the range, the opcode for jump is 2 or 000010 and the target is 0xFFFFFA8 >> 2 = 0x3FFFEA

14. (10 points) Assume for arithmetic, load/store, and branch instructions, a processor has CPIs of 1, 6, and 3, respectively. Also assume that on a single processor a program requires the execution of 3.5 E9 arithmetic instructions, 1.2 E9 load/store instructions, and 200 E6 branch instructions. Assume that each processor has a 2 GHz clock frequency.

Assume that, as the program is parallelized to run over multiple cores, the number of arithmetic and load/store instructions per processor is divided by 0.85 x p (where p is the number of processors) but the number of branch instructions per processor remains the same. Find the total execution time for this program on 1, 2, 4, and 8 processors, and show the relative speedup of the 2, 4, and 8 processor result relative to the single processor result.

$$ET_{orig} = \frac{3.5E9*1+1.2E9*6+200\,E6*3}{2E9} = 5.65\ s$$

$$ET_{p=2} = \frac{\frac{3.5E9}{2*0.85}*1+\frac{1.2\,E9}{2*0.85}*6+200\,E6*3}{2E9} = 3.45\ s,\ Speedup = \frac{ET_{orig}}{ET_{p=2}} = \frac{5.65}{3.45} = 1.64$$

$$ET_{p=4} = \frac{\frac{3.5E9}{4*0.85}*1+\frac{1.2\,E9}{4*0.85}*6+200\,E6*3}{2E9} = 1.87\ s,\ Speedup = \frac{ET_{orig}}{ET_{p=4}} = \frac{5.65}{1.87} = 3.02$$

$$ET_{p=8} = \frac{\frac{3.5E9}{8*0.85}*1+\frac{1.2\,E9}{8*0.85}*6+200\,E6*3}{2E9} = 1.09\ s,\ Speedup = \frac{ET_{orig}}{ET_{p=8}} = \frac{5.65}{1.09} = 5.18$$

15. (6 points) The following list provides parameters of a virtual memory system.

Virtual Address (bits)	Physical DRAM Installed	Page Size	PTE Size (byte)
48	32 GiB	16 KiB	8

a. (2 points) For a single-level page table, how many page table entries (PTEs) are needed?
#VP = $2^{48}/2^{14} = 2^{34}$ page table entries

b. (4 points) How much physical memory is needed for storing the page table?
#entries x PTE size = 2^{34} x 2^{3} = 128 GiB

16. (10 points) Consider the NIOS iiF embedded processor found on Intel FPGA chips. There are three NIOS instruction word formats: I-type, R-type, and J-type which have the following characteristics:
I-Type: 6-bit opcode, two 5 bit register fields A and B, 16-bit immediate data field IMM16
R-Type: 6-bit opcode, three 5-bit register fields A, B, C, 11-bit opcode-extension field OPX
J-Type: 6-bit opcode, 26-bit immediate data field

`add rC, rA, rB`	rC ← rA + rB
`addi rB, rA, IMM16`	rB ← rA + Signextend(IMM16)
`beq rA, rB, label`	if (rA == rB) then PC ← PC + 4 + Signextend(IMM16) else PC ← PC + 4
`ldw  rB, byte_offset(rA)`	rB ← Mem32[rA + Signextend(IMM16)]
`mul rC, rA, rB`	rC ← (rA x rB)$_{31..0}$
`ror rC, rA, rB`	rC ← rA rotated right rB$_{4..0}$ bit positions

This processor has a six stage pipeline as shown below. F, D, E, M and W are similar to the MIPS pipeline stages and the Align stage is particular to the Avalon memory management unit.

Stage Letter	Stage Name
F	Fetch
D	Decode
E	Execute
M	Memory
A	Align
W	Writeback

Instruction	Cycles	Penalties
Normal ALU instructions (e.g., add, cmplt)	1	
Branch (correctly predicted, taken)	2	
Branch (correctly predicted, not taken)	1	
Branch (mispredicted)	4	Pipeline flush
jmp, ret, callr	3	
Load	1	Late result
Store	1	
Shift/rotate	1	Late result

A D-stage stall occurs if an instruction is trying to use the result of a late result instruction too early. Late result instructions have two cycles placed between them and an instruction that uses their result. Instructions that flush the pipeline cause up to three instructions after them to be cancelled. This creates a three-cycle and an execution time of four cycles.

How many cycles will it take to execute the following NIOS code if the beq is mispredicted?

```
a      add    r5, r7, r9                  1
b      lw     r9, 400(r5)                 1 + 2 late result r5
c      ror    r9, r9, r2                  1 + 2 late result r9
d      sw     r9, 200(r3)                 1
e      beq    r5, r5, labelit             4 mispredicted branch
f      addi   r5, r5, 20                  not executed because branch is
                                          taken, r5 is equal to r5
Total                                     12 cycles
```

Dependences: a-b, a-e, b-c, c-d