The University of Alabama in Huntsville
ECE Department
CPE 431 01
Test 1 Solution
Fall 2016

Remember: ET = IC*CPI*CT

1. (1 point) _Amdahl's law_ states that the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used.

2. (1 point) The layout of the instruction is called the _instruction format_.

3. (1 point) _Benchmarks_ are programs specifically chosen to measure performance.

4. (1 point) A _return address_ is a link to the calling site that allows a procedure to return to the proper address; in MIPS it is stored in register `$ra`.

5. (1 point) _Overflow_ occurs when a number can't be represented in a computer.

6. (10 points) MIPS chooses to simplify the structure of its instructions. It implements complex instructions by decomposing them into multiple simpler MIPS ones. The complex instruction `swap $rs, $rt` can be decomposed into three instructions. If the implementation of the swap instruction in hardware will increase the clock period of a single-cycle implementation by 10%, what percentage of swap operations in the instruction mix would recommend implementing it in hardware?

   Let x represent the proportion of instructions that are swap instructions. Consider two cases: (1) before change in which swap instructions are pseudo-instructions which are replaced by three instructions, (2) after change in which swap instructions are implemented directly in hardware.

   $CPI_{before} = CPI_{after} = 1$, $CT_{after} = 1.1CT_{before}$, $IC_{before} = (x*3 +(1-x)*1)IC_{after}$

   $ET_{before} = ET_{after}$

   $CPI_{before}*IC_{before}*CT_{before} = CPI_{after}*IC_{after}*CT_{after}$

   Substituting for IC_{before} and CT_{after},,

   $1*(x*3 +(1-x)*1)IC_{after}*CT_{before} = 1*IC_{after}*1.1CT_{before}$

   $3x + 1 – x = 1.1$

   $2x = 0.1$

   $x = 0.05$ or 5 %

7. (15 points) Translate the following C code to MIPS. Assume that the variables `f`, `g`, `h`, `i` and `j` are given and 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. Assume that the elements of the arrays A and B are 4-byte words:

```
B[8] = A[i + j]
```

```
add     $t0, $s3, #s4
sll     $t0, %t0, 2
add     $t0, $t0, $s6
lw      $t0, 0($t0)
sw      $t0, 32($s7)
```

8. (15 points) Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes of instructions according to their CPI (class A, B, C, and D). P1 with a clock rate of 2.75 GHz and CPIs of 1, 2, 2, and 3, and P2 with a clock rate of 3 GHz and CPIs of 1, 1, 2, and 3.

Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 15% class A, 20% class B, 45% class C, and 20% class D, which implementation is faster? What clock rate must be achieved by the slower processor to make it as fast as the other with no other changes?

$CPI_{P1} = 0.15 * 1 + 0.2 * 2 + 0.45 * 2 + 0.2 * 3 = 0.15 + 0.4 + 0.9 + 0.6 = 2.05$
$CPI_{P2} = 0.15 * 1 + 0.2 * 1 + 0.45 * 2 + 0.2 * 3 = 0.15 + 0.2 + 0.9 + 0.6 = 1.85$

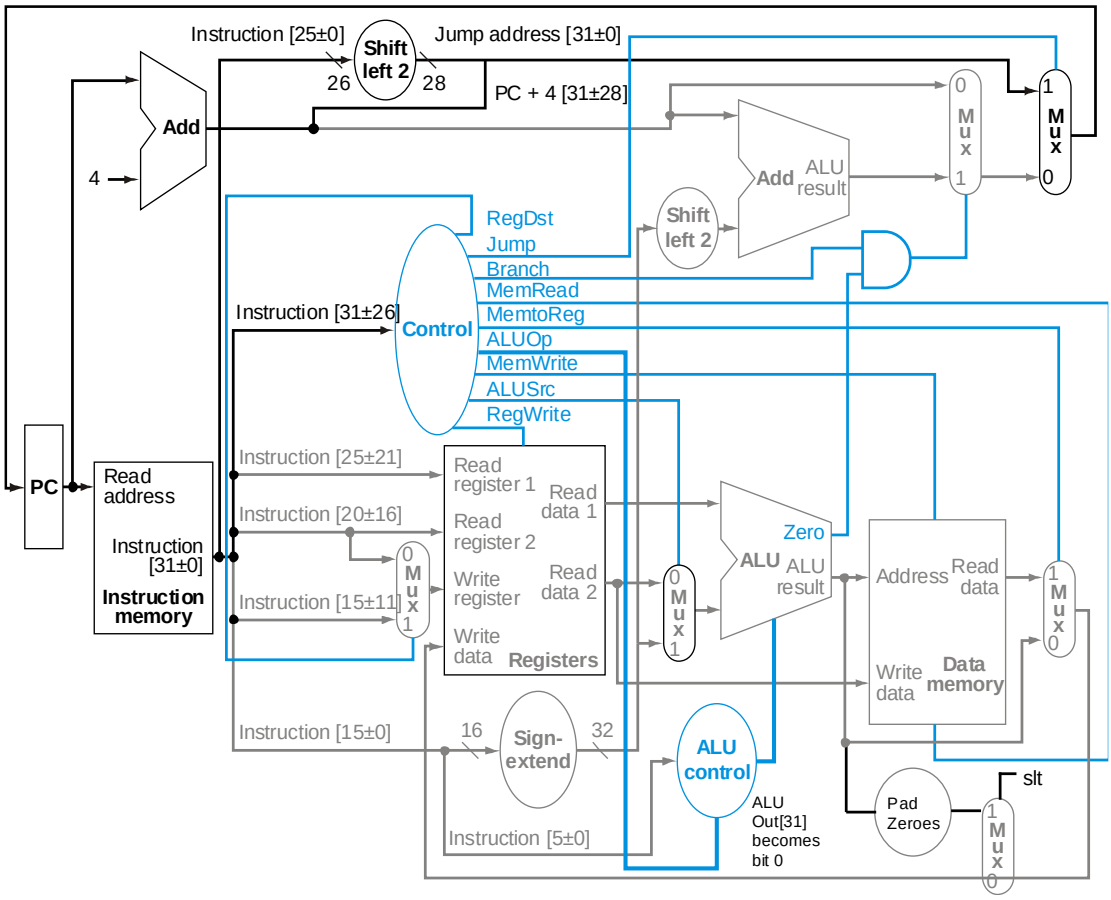
$$\frac{P_{P1}}{P_{P2}} = \frac{ET_{P2}}{ET_{P1}} = \frac{\frac{IC_{P2}CPI_{P2}}{CR_{P2}}}{\frac{IC_{P1}CPI_{P1}}{CR_{P1}}} = \frac{CPI_{P2}CR_{P1}}{CR_{P1}CPI_{P2}} = \frac{1.85*2.75}{2.05*3} = 0.827$$

P2 is faster by 1.209

$$\frac{1.85*CR_{P1}}{2.05*3} = 1$$

CR_{P1} = 3.32 GHz

9. (15 points) Add the instruction `slt` to the single-cycle datapath shown in the figure below. The begz instruction is defined below. Add any necessary datapaths and control signals and show the necessary additions to the table of control signals given.



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

d-don't care

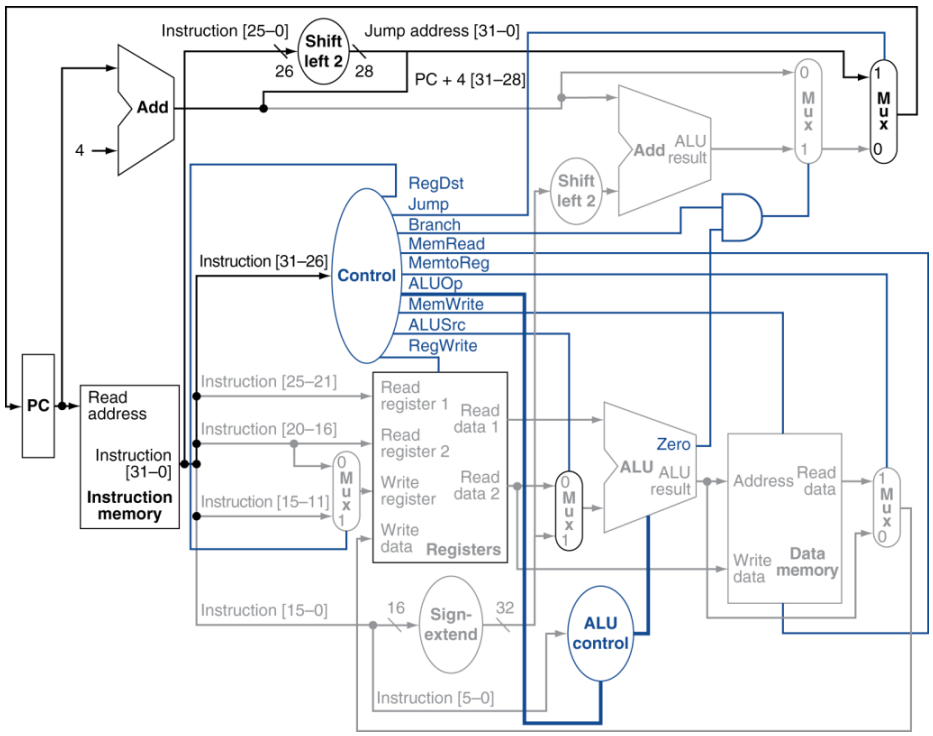
10. (10 points) A MIPS processor fetches the following instruction word:

`0000 0000 0000 0010 0000 1010 1000 0000`

Assume that data memory is all zeros and that the processor's registers have the following values at the beginning of the cycle in which the above instruction word is fetched.

r0	r1	r2	r3	r4	r5	r6	r8	r12	r31
0	-1	2	-3	-4	10	6	8	2	-16

What instruction is this and what is the output of the sign-extend for this instruction word?



`0000 00_00 000_0 0010 0000 1_010 10_00 0000`

opcode = 000000, R-type

rs = 00000, $zero

rt = 00010, $v0

rd = 00001, $at

shamt = 01010

funct = 000000, sll

R[rd] ← R[rt] << shamt

`sll $at, $v0, 10`

Instruction [16..0] = 0000 1010 1000 0000

Sign Extend = 0x0000 0A80

11. (20 points) In this exercise, we assume that the following MIPS code is executed on a pipelined processor with a 5-stage pipeline, full forwarding, and a predict-taken branch predictor. Draw the pipeline execution diagram for this code, assuming there are no delay slots and that branches execute in the EX stage.

```
          add   $t1, $t1, $t0
          lw    $t2, 0($t1)
label1:   beq   $t2, $zero, label2    # not taken once, then taken
          lw    $t3, 0($t2)
          sw    $t3, 0($t1)
          beq   $t3, $zero, label1    # taken
          add   $t1, $t3, $t1
label2:   sw    $t1, 0($t2)
```

Cycle	IF	ID	EX	MEM	WB
1	add				
2	lw	add			
3	beq	lw	add		
4	sw	beq	lw	add	
5	sw	beq	bubble	lw	add
6	after	sw	beq	bubble	lw
7	lw $t3	bubble	bubble	beq	bubble
8	sw $t3	lw $t3	bubble	bubble	beq $t2
9	beq $t3	sw $t3	lw $t3	bubble	bubble
10	beq $t3	sw $t3	bubble	lw $t3	bubble
11	beq $t2	beq $t3	sw $t3	bubble	lw $t3
12	sw $t1	beq $t2	beq $t3	sw $t3	bubble
13		sw $t1	beq $t2	beq $t3	sw $t3
14			sw $t1	beq $t2	beq $t3
15				sw $t1	beq $t2
16					sw $t1

12. Write down the hexadecimal representation of the decimal number 1297.71875 assuming the IEEE 754 single precision format.

```
/2       0   1            x2    0.71875
/2       1   0            x2    1.4375       1
/2       2   1            x2    0.875        0
/2       5   0            x2    1.75         1
/2      10   0            x2    1.5          1
/2      20   0            x2    1.0          1
/2      40   1
/2      81   0
/2     162   0
/2     324   0
/2     648   1
/2    1297
```

101 0001 0001.1011 1 = 1.0100 0100 0110 111 x 2^{10}
S = 0, EXP + BIAS = 10 + 127 = 137 = 1000 1001
Fraction = 0100 0100 0110 111
0100 0100 1010 0010 0011 0111 0000 0000 = 0x44A2 3700