The University of Alabama in Huntsville
Electrical & Computer Engineering Department
CPE 431 01
Test 1 Solution
Fall 2011

1. (1 point) A number in floating-point notation that has no leading 0s is said to be __normalized_.
2. (1 point) The address specified in a branch, which becomes the new program counter if the branch is taken is known as the __target or destination_ address.
3. (1 point) The _caller_ .instigates a procedure and provides the necessary parameter values.
4. (1 point) An __embedded system__.is a computer inside another device used for running one predetermined application or collection of software.
5. (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.
6. (9 points) For the MIPS assembly below, assume that the registers `$s0`, `$s1`, `$s2`, `$s3`, contain the values 10, 20, 30, and 40, respectively. Also, assume that register `$s6` contains the value 256, and that memory contains the following values:

Address	Value
256	100
260	200
264	300

```
(1)   addi   $s6, $s6, -20
(2)   add    $s6, $s6, $s1
(3)   lw     $s0, 8($s6)
```

Find the value of $s0 at the end of the assembly code.

Initially, `$s6` contains 256 and `$s1` contains 20. After the execution of (1), `$s6` contains 256 + (-20) = 236. After the execution of (2), `$s6` contains 236 + 20 = 256, After the execution of (3), `$s6` has MEM[8+256] = 300.

7. (6 points) What are the binary representations of the opcode, rs, rt, rd, shamt, and funct fields in this instruction?

```
addi $a1, $v0, -6
```

`addi $a1, $v0, -6` is an I- type instruction so has the fields opcode, rs, rt, and Offset/Immediate.

The opcode for addi is 8_{10}, 001000
rs (`$v0`) is 2 = 00010
rt (`$a1`) is 5 = 00101
-6 = -(+6) = -(0000 0000 0000 0110) = 1111 1111 1111 1010

8. (5 points) Show the IEEE 754 binary representation for the floating-point number 307.75_{10} in single precision.

$307.75_{10} = 100110011.11_2 = 1.0011001111 \times 2^8$
Sign = 0, positive number
Exponent = $8 + 127 = 135_{10} = 10000111_2$
Fraction = 001 1001 1110 0000 0000 0000

Putting them together 0100 0111 1001 1001 1110 0000 0000 0000 = 0x4399 E000

```
2      0    1                0.75    2
2      1    0                1.50    2
2      2    0                1.00
2      4    1
2      9    1
2     19    0
2     38    0
2     76    1
2    153    1
2    307
```

9. (15 points) Consider three different P1, P2, and P3 executing the same instruction set with the clock rates and CPIs given in the following table.

Processor	Clock Rate	CPI
P1	2 GHz	1.3
P2	1.4 GHz	1.0
P3	2.6 GHz	2.2

Which processor has the highest performance?

$IC_1 = IC_2 = IC_3 = IC$

$$P = \frac{1}{ET} = \frac{1}{IC \times CPI \times CT} = \frac{CR}{IC \times CPI}$$

$$P_1 = \frac{1}{ET_1} = \frac{CR_1}{IC_1 \times CPI_1} = \frac{2 \times 10^9 \, cycles/s}{IC \times 1.3 cycles/instruction} = \frac{1.54 \times 10^9}{IC}$$

$$P_2 = \frac{1}{ET_2} = \frac{CR_2}{IC_2 \times CPI_2} = \frac{1.4 \times 10^9 \, cycles/s}{IC \times 1.0 cycles/instruction} = \frac{1.4 \times 10^9}{IC}$$

$$P_3 = \frac{1}{ET_3} = \frac{CR_1}{IC_3 \times CPI_3} = \frac{2.6 \times 10^9 \, cycles/s}{IC \times 2.2 cycles/instruction} = \frac{1.18 \times 10^9}{IC}$$

$P_1 > P_2 > P_3$, so P_1 has the highest performance.

10. (5 points) Suppose the program counter (PC) is set to 0x0000 0020. Is it possible to use the jump (j) MIPS assembly instruction to set the PC to the address 0x0000 0021?

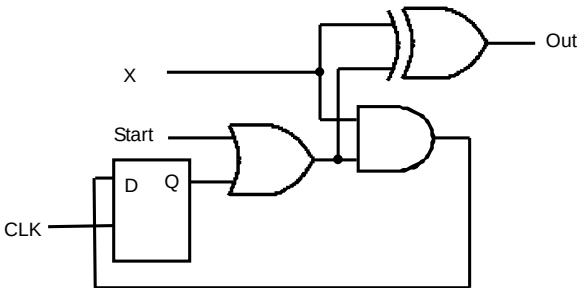
The range of the jump is limited by the four highest order bits of the program counter, in this case 0000. The range of addresses is 0x0000 0000 to 0x0FFF FFFC. The last two bits are always 0, the 26 bits of the jump instruction is shifted left by two and concatenated with the upper 4 bits of the PC. 0x0000 0021 is in range for this instruction but not on a word boundary, so, no it is not possible.

11. (10 points) What decimal number does the bit pattern represent if it is a two's-complement integer? An unsigned integer?

1011 1100 0010 0101

Signed: $-2^{15} + 2^{13} + 2^{12} + 2^{11} + 2^{10} + 2^5 + 2^2 + 2^0 = -17371_{10}$
Unsigned: $2^{15} + 2^{13} + 2^{12} + 2^{11} + 2^{10} + 2^5 + 2^2 + 2^0 = 48165_{10}$

12. (10 points) Consider the following circuit:



Assume that the latency and cost of basic logic elements are as follows:

NOT		AND		OR		XOR		D-element	
Latency	Cost	Latency	Cost	Latency	Cost	Latency	Cost	Latency	Cost
40 ps	1	50 ps	2	60 ps	2	80 ps	3	80 ps	12

The time given for a D-element is its setup time. The data input of a flip-flop must have the correct value one setup-time before the clock edge (end of clock cycle) that stores that value into the flip-flop. What is the cycle time for the circuit given?
Paths to consider:

Start to Out: OR + XOR = 60 ps + 80 ps = 140 ps
X to Out: XOR = 80 ps
Start to Input of Flip-Flop: OR + AND + D-element= 60 ps + 50 ps + 80 ps= 190 ps
X to Input of Flip-Flop: AND + D-element = 60 ps + 80 ps = 110 ps
Output of FF to Out: OR + XOR = 60 ps + 80 ps = 140 ps
Output of FF to Input of FF: OR + AND + D-element = 60 ps + 50 ps + 80 ps = 190 ps

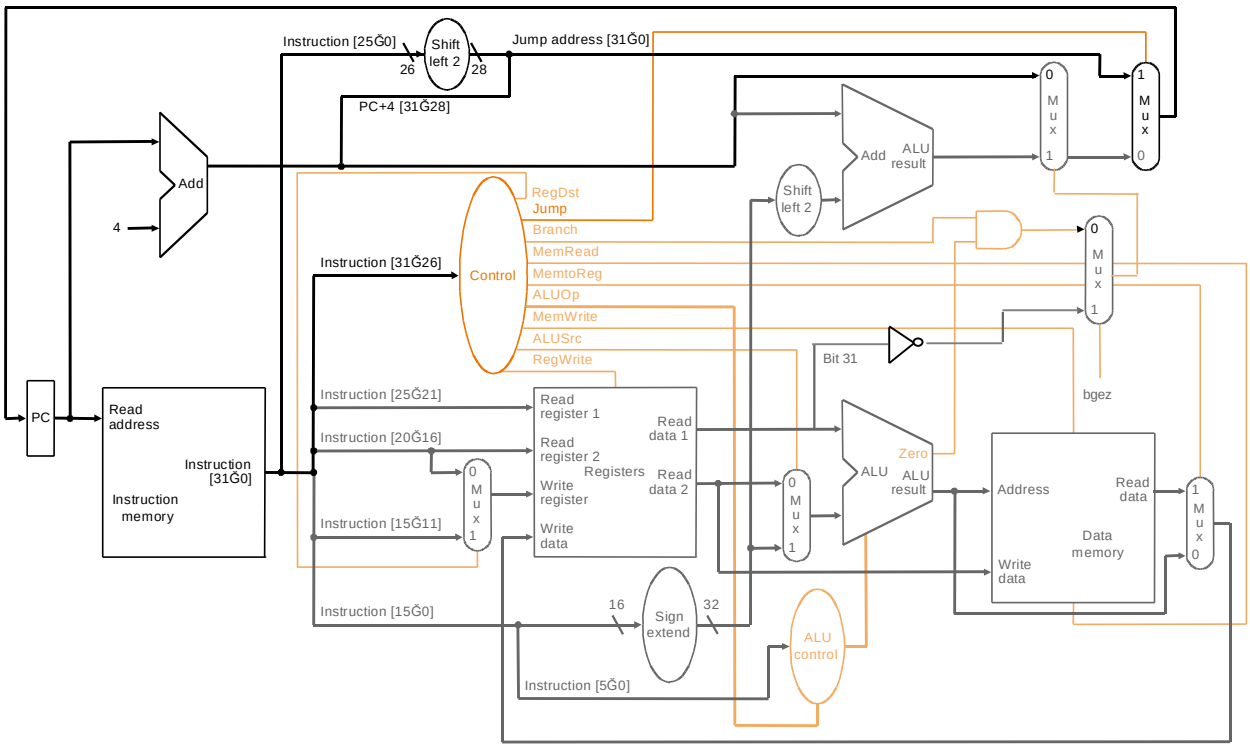
The maximum is 190 ps

13. (15 points) Add the instruction `bgez` (branch on greater than or equal to zero) 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.

`begz rs, label`

if (rs >= 0)
    PC ← PC + 4 + 4*offset
else
    PC ← PC + 4

1	rs	1	offset



Instruction	RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0	bgez
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
bgez	d	d	d	0	0	0	d	d	d	1

d-don't care

14. (10 points) Assume the following CPIs and the instruction breakdown for executing a given program:

	Instructions (in millions)	CPI
Arithmetic	500	4
Load/Store	300	10
Branch	100	2

Suppose that new, more powerful arithmetic instructions are added to the instruction set. On average, through the use of these more powerful arithmetic instructions, we can reduce the number of arithmetic instructions need to execute a program by 20 %, and the cost of increasing the clock cycle time by only 15 %. Is this a good design choice? Why?

$CT_{new} = 1.15*CT_{old}$
$CPI_{old} = (500/900)*4 + (300/900)*10 + (100/900)*2$
$CPI_{new} = (400/800)*4 + (300/800)*10 + (100/800)*2$

$$\frac{P_{new}}{P_{old}} = \frac{ET_{old}}{ET_{new}} = \frac{IC_{old} \times CPI_{old} \times CT_{old}}{IC_{new} \times CPI_{new} \times CT_{new}} = \frac{900 \times 10^6 \times \frac{2000 + 3000 + 200}{900} \times CT_{old}}{800 \times 10^6 \times \frac{1600 + 3000 + 200}{800} \times 1.15 \times CT_{old}} = 0.942$$

The performance of the modified instruction set is worse than the original, so no, this is not a good design choice.

15. (10 points) Your company could speed up a Java program on their new computer by adding hardware support for garbage collection. Garbage collection currently comprises 20% of the cycles of the program. You have two possible changes to the machine. The first one would be to automatically handle garbage collection in hardware. This causes an increase in cycle time by a factor of 1.2. The second would be to provide for new hardware instructions to be added to the ISA that could be used during garbage collection. This would halve the number of instructions needed for garbage collections but increase the cycle time by 1.1. Which of these two options, if either, should you choose?

Option 1: $CT_{new} = 1.2*CT_{old}$
$CC_{new} = 0.8*CC_{old}$

Option2: $CT_{new} = 1.1*CT_{old}$
$IC_{gc\_new} = 0.5*IC_{gc\_old}$
$IC_{gc\_old} = (CCgc\_old/CPIgc\_old) = 0.2CC_{old}/CPI_{gc\_old}$
$CC_{gc\_new} = 0.5*IC_{gc\_old}*CPI_{gc\_new}$
Assume $CPI_{gc\_new} = CPI_{gc\_old}$
$CC_{gc\_new} = 0.5*(0.2*CC_{old}/CPI_{gc\_old})*CPI_{gc\_new}$
$= 0.1* CC_{old}$
$CC_{new} = 0.9*CC_{old}$

$$\frac{P_{Option\ 1}}{P_{old}} = \frac{ET_{old}}{ET_{Option\ 1}} = \frac{CC_{old} \times CT_{old}}{CC_{Option\ 1} \times CT_{Option\ 1}} = \frac{CC_{old} \times CT_{old}}{0.8 \times CC_{old} \times 1.2 \times CT_{old}} = \frac{1}{0.96} = 1.04$$

$$\frac{P_{Option\ 2}}{P_{old}} = \frac{ET_{old}}{ET_{Option\ 2}} = \frac{CC_{old} \times CT_{old}}{CC_{Option\ 2} \times CT_{Option\ 2}} = \frac{CC_{old} \times CT_{old}}{0.9 \times CC_{old} \times 1.1 \times CT_{old}} = \frac{1}{0.99} = 1.01$$

Both option 1 and option 2 are improvements, 1 is more of an improvement than 2.