**The University of Alabama in Huntsville**
**Electrical & Computer Engineering Department**
**CPE 431 01**
**Fall 2003**
**Final Exam Solution**

1. (10 points) We are interested in two implementations of a machine, one with and another without, special floating-point hardware. Consider a program, P, with the following mix of operations:

| | |
|---|---|
| floating-point multiply | 10 % |
| floating-point add | 15 % |
| floating-point divide | 5 % |
| integer instructions | 70 % |

Machine MFP (Machine with Floating Point) has floating-point hardware and can therefore implement the floating-point operations directly. It requires the following number of clock cycles for each instruction class:

| | |
|---|---|
| floating-point multiply | 6 |
| floating-point add | 4 |
| floating-point divide | 20 |
| integer instructions | 2 |

Machine MNFP (Machine with No Floating Point) has no floating-point hardware and so must emulate the floating-point operations using integer instructions. The integer instructions all take 2 clock cycles. The number of integer instructions needed to implement each of the floating-point operations is as follows: Both machines have a clock rate of 1000 MHz.

| | |
|---|---|
| floating-point multiply | 30 |
| floating-point add | 20 |
| floating-point divide | 50 |

If the machine MFP needs 300 million instructions for this program, how many integer instructions does the machine MNFP require for the same program?

$$IC_{MFP} = 300 \times 10^6$$

$$IC_{MNFP} = 300 \times 10^6 (0.7 * 1 + 0.1 * 30 + 0.15 * 20 + 0.05 * 50) = 300 \times 10^6 (0.7 + 3.0 + 3.0 + 2.5)$$

$$= 300 \times 10^6 * 8.2 = 2.46 \times 10^9$$

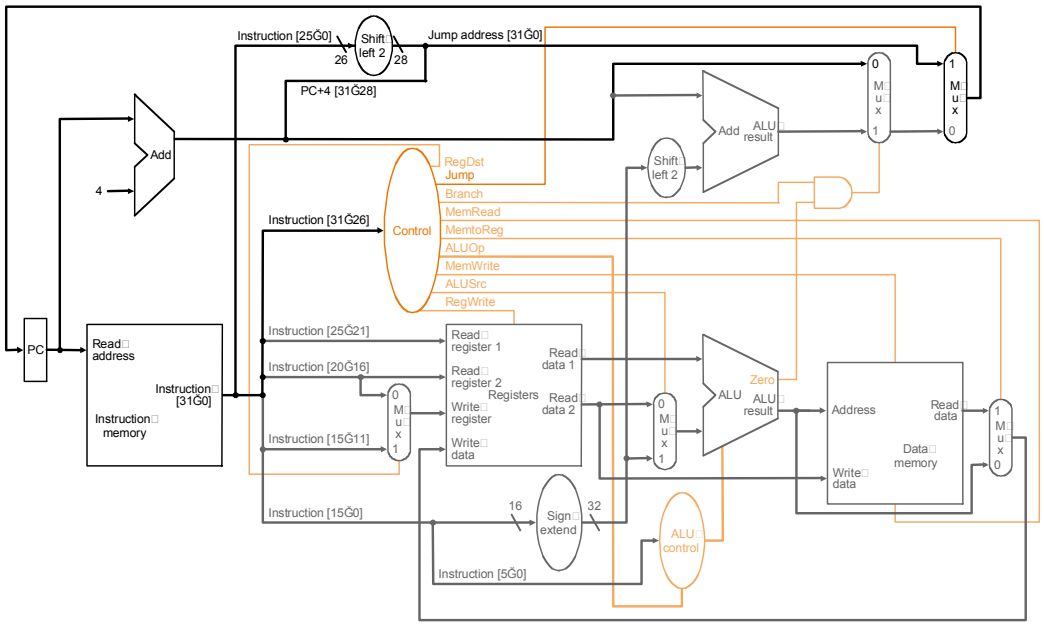
2. (1 point) _____Bandwidth_____ is a term for the amount of information delivered per second.

3. (10 points) Add a variant of the instruction `lw` instruction, which sums two registers to obtain the address of the data to be loaded and uses the R-format. to the single-cycle datapath shown in the figure below. Add any necessary datapaths and control signals and show the necessary additions to the table of control signals given.

no modifications to the datapath are required

| Instruction | RegDst | ALUSrc | Memto Reg | Reg Write | Mem Read | Mem Write | Branch | ALUOp1 | ALUOp0 | |
|---|---|---|---|---|---|---|---|---|---|---|
| R-format | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | |
| lw | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | |
| sw | d | 1 | d | 0 | 0 | 1 | 0 | 0 | 0 | |
| beq | d | 0 | d | 0 | 0 | 0 | 1 | 0 | 1 | |
| lw(variant) | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | |

d-don't care

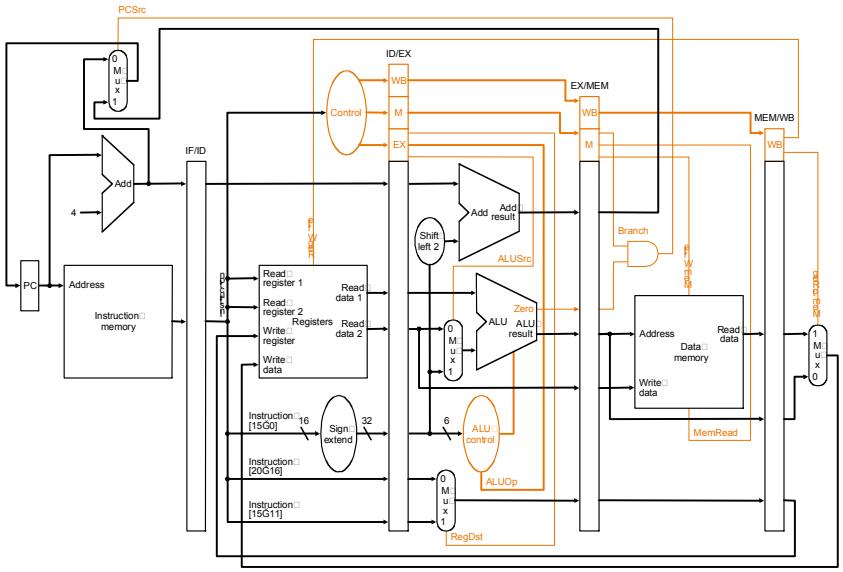



4. (1 point) A ___bus____ is a shared communication link, which uses one set of wires to connect multiple subsystems.

5. (1 point) ___Communication___ is the hardest problem.

6. (2 points) Many times in computer architecture, a person must consider the tradeoff between ____space__ and ___time___.

7. (15 points) Consider executing the following code on a pipelined datapath like the one shown, which has no forwarding and in which branches complete in the MEM stage:



```
        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
500     for1tst: slt $t0, $s0, $s3
504              beq $t0, $zero, exit1
508              addi$s1, $s0, -1
512     for2tst: slt $t0,$s1, $zero
516              bne $t0, $zero, exit2
520              add $t1, $s1, $s1
524              add $t1, $t1, $t1
528              add $t2, $s2, $t1
532              lw  $t3, 0($t2)
536              lw  $t4, 4($t2)
540              slt $t0, $t4, $t3
                 beq $t0, $zero, exit2
                 add $a0, $s2, $zero
                 add $a1, $s1, $zero
                 jal swap
                 addi$s1, $s1, -1
                 j   for2tst
             exit2:  addi$s0, $s0, 1
                 j   for1tst
             exit1:  lw  $ra, 16($sp)
                 lw  $s0, 0($sp)
                 lw  $s1, 4($sp)
                 lw  $s2, 8($sp)
                 lw  $s3, 12($sp)
                 addi$sp, $sp, 20
                 jr  $ra
```

If the `slt $t0` instruction at the `for1tst` label begins executing in cycle 1 and the `beq $t0, $zero, exit1` is not taken and the `bne $t0, $zero, exit2` is taken, what are the values stored in the following fields of the IF/ID pipeline register in the 15th cycle? Assume that before the instructions are executed, the state of the machine was as follows:

The PC has the value $500_{10}$, the address of `slt $t0` (at the label `for1tst`).

Every register has the initial value $20_{10}$ plus the register number.

Every memory word accessed as data has the initial value $2000_{10}$ plus the byte address of the word.

Fill in all of the fields, even if the current instruction in that state is not using them.

The instruction of interest is `add $t2, $s2, $t1`.

IF/ID.PCInc = 532
IF/ID.Instruction = $02495020_{16}$

| Cycle | IF | ID | EX | MEM | WB |
|---|---|---|---|---|---|
| 1 | slt $t0 | | | | |
| 2 | beq $t0 | slt $t0 | | | |
| 3 | addi $s1 | beq $t0 | slt $t0 | | |
| 4 | addi $s1 | beq $t0 | bubble | slt $t0 | |
| 5 | addi $s1 | beq $t0 | bubble | bubble | slt $t0 |
| 6 | slt $t0 | addi $s1 | beq $t0 | bubble | bubble |
| 7 | bne $t0 | slt $t0 | addi $s1 | beq $t0 | bubble |
| 8 | bne $t0 | slt $t0 | bubble | addi $s1 | beq $t0 |
| 9 | bne $t0 | slt $t0 | bubble | bubble | addi $s1 |
| 10 | add $t1, $s1 | bne $t0 | slt $t0 | bubble | bubble |
| 11 | add $t1, $s1 | bne $t0 | bubble | slt $t0 | bubble |
| 12 | add $t1, $s1 | bne $t0 | bubble | bubble | slt $t0 |
| 13 | add $t1, $t1 | add $t1, $s1 | bne $t0 | bubble | bubble |
| 14 | add $t2 | add $t1, $t1 | add $t1, $s1 | bne $t0 | bubble |
| 15 | addi $s0 | bubble | bubble | bubble | bne $t0 |

8. (5 points) Show the single MIPS instruction or minimal sequence of instructions for this C statement:

```
x[10] = a + c;
```

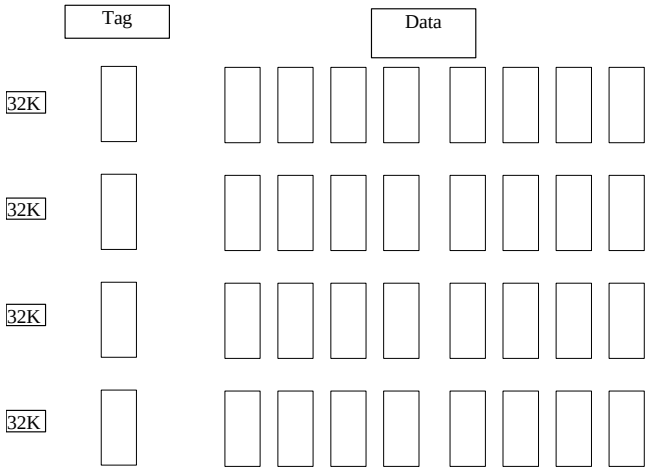
Assume that `a` and `c` correspond to registers `$t0 and t1,` respectively, and the array `x` has a base address of $4{,}000{,}000_{10}$ which is stored in register `$t2`.

```
add   $t1, $t0, $t1
sw    $t1, 40($t2)
```

9. (15 points) You have been given 50 32K x 16-bit SRAMS to build an instruction cache for a processor with a 32-bit address. You do not have a byte offset. What is the largest size (i.e., the largest size of the data storage area in bytes) direct-mapped instruction cache that you can build with four-word blocks? Show the breakdown on the address into its cache access components and describe how the various SRAM chips will be used.

Byte offset is 0 bits, Block offset is 2 bits, Index n bits, Tag 32 - n - 2. It takes 8 chips to get four words, and we have increments of 128K words. 128K words comes from 32K sets, for which 15 bits of index are required, leaving 15 bits of tag. 15 bits for each tag requires 1 chip. The data requires 8 chips (2 chips/word, 4 words/block, 1 block per set). Tag plus data for 128K words requires 9 total chips. We can quadruple this and still have extra chips. So, the total data storage is 128K sets * 4 words/set *4 bytes/word for a total data storage of 2MB and the address is broken down as

Byte offset – 0 bits
Block offset – 2 bits
Index – 17 bits
Tag – 13 bits




10. (10 points) A program repeatedly performs a three-step process: It reads in a 4-KB block of data from another disk, does some processing on that data, and then writes out the result as another 4-KB block elsewhere on the disk. Each block is contiguous and randomly located on a single track on the disk. The disk drive rotates at 7200 RPM, has an average seek time of 8 ms, and has a transfer rate of 20 MB/Sec. The controller overhead is 2 ms. No other program is using the disk or processor, and there is no overlapping of disk operation with processing. The processing takes 20 million clock cycles, and the clock rate is 400 MHz. What is the overall speed of the system in blocks processed per second?

Read/Write = controller + seek + rotational + transfer

$$= 2\text{ ms} + 8\text{ ms} + 0.5 rotation * \frac{1\min}{7200 rev} * \frac{60 s}{1\min} + \frac{4 KB}{20 MB/s}$$

$$= 10\text{ ms} + 4.17\text{ ms} + 0.2\text{ ms} = 14.37\text{ ms}$$

$$\text{Processing} = 20 \times 10^6 cycles * \frac{1 s}{400 \times 10^6 cycles} = 50 ms$$

Block time = read + processing + write

$$= (14.37 + 50 + 14.37)\text{ ms} = 78.74\text{ ms}$$

$$\text{Blocks processed per second} = \frac{1 block}{78.74 ms} = 12.7 blocks/s$$

11. (15 points) Suppose we have a system with the following characteristics:
    a. A memory and bus system supporting block access of 4 to 16 32-bit words.
    b. A 64-bit synchronous bus clocked at 100 MHz, with each 64-bit transfer taking 1 clock cycle, and 1 clock cycle required to send an address to memory.
    c. Three clock cycles needed between each bus operation. (Assume the bus is idle before an access.)
    d. A memory access time for the first four words of 400 ns; each additional set of four words can be read in 60 ns. Assume that a bus transfer of the most recently read data and a read of the next four words can be overlapped.

    Consider a write-back cache used for a processor with a bus and memory system as described above (assume that writes require the same amount of time as reads). The following performance measurements have been made:

    The cache miss rate is 0.05 misses per instruction for block sizes of 8 words.
    The cache miss rate is 0.03 misses per instruction for block sizes of 16 words.
    For either block size, 40% of the misses require a write-back operation, while the other 60% require only a read.

    Assuming that the processor is stalled for the duration of a miss (including the write-back time if a write-back is needed), find the number of cycles per instruction that are spent handling cache misses for each block size.

**8 word blocks**

| | | | | | | |
|---|---|---|---|---|---|---|
| **Bus** | **1 send** | **idle** | **2 send** | **4 idle** | **2 send** | **3 idle** |
| **Memory** | **idle** | **40 cycles read** | **6 cycles read** | | | |

**Total = 52 cycles**

**16 word blocks**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Bus** | **1 send** | **idle** | **2 send** | **4 idle** | **2 send** | **4 idle** | **2 send** | **4 idle** | **2 send** | **3 idle** |
| **Memory** | **idle** | **40 cycles read** | **6 cycles read** | | **6 cycles read** | | **6 cycles read** | | **5 cycles idle** | |

**Total =64 cycles**

8 words (0.6 * 52 cycles + 0.4 * 104 cycles) * 0.05 = (31.2 + 41.6) * 0.05 = 72.8 * 0.05 = 3.64
16 words (0.6 * 64 cycles + 0.4 * 128 cycles) * 0.03 = (38.4 + 51.2) * 0.03 = 89.6 * 0.03 = 2.69

12. (5 points). Given the bit pattern:

```
1000 1101 1110 1001 1100 0000 0010 0000
```

what does it represent, assuming that it is a single precision floating-point number?

Since the most significant bit is 1, this is a negative number. The exponent is 0001 1011 minus the bias of 0111 1111 (127) which is –100. The rest of the number is the mantissa. The number is $-(1 + 2^{-1} + 2^{-2} + 2^{-4} + 2^{-7} + 2^{-8} + 2^{-9} + 2^{-18})*2^{-100}$

13. (10 points) In this exercise, we'll examine quantitatively the pros and cons of 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 | New Frequency |
|---|---|---|
| Arithmetic | 48 % | 48 % |
| Data Transfer | 33 % | 33 % (0.4) + 33 % (0.6)(1-$x^*$) |
| Conditional branch | 17 % | 17 % |
| Jump | 2 % | 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,

$ET_{orig} = ET_{mod}$

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

$IC_{orig} = IC_{mod} * 1.15$

Icorig = Icorig(1-0.198x) * 1.15

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

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

x = 65.9