CS 352 Computer Architecture
Exam #1 – Prof. Mark
March 9, 2004

Name: Key
(please print)

1-2.	9 points	_____
3-5.	24 points	_____
6.	20 points	_____
7.	12 points	_____
8.	15 points	_____
9.	20 points	_____
	TOTAL (100 pts)	_____

In recognition of University regulations regarding academic dishonesty, I certify that I have neither given nor received unpermitted aid on this examination. One sheet of notes (one side of 8.5 x 11 paper) and a non-programmable calculator are allowed.

Signed: _____

HW #1	HW #2	HW #3	HW #4	HW #5

(leave blank)

Multiple Choice

1. Which of the following would be changes to the architecture (A) of a MIPS computer and which would be changes to its implementation (I)? Circle A or I to indicate your choice. (4 points)

A (I)	Increasing the clock rate, while changing nothing else
(A) I	Adding an extra 16 general-purpose integer registers
A (I)	Adding an instruction cache
(A) I	Expanding the instruction size from 32 bits to 64 bits
A (I)	Adding an extra stage in the pipeline
(A) I	Eliminating the branch delay slot
A (I)	Removing some of the forwarding/bypassing capabilities from the data path
(A) I	Adding support for quad-precision floating point

2. Which is true about a pipelined processor?
(Circle **ALL** correct responses – there is more than one for this question)
(5 points)

(a.) Adding more pipeline stages usually increases the number of control and data hazards

(b.) Adding more pipeline stages is one good way to increase the clock rate of a processor.

c. Latch overheads do not limit the maximum number of stages in a pipeline

d. Changing the number of pipeline stages typically requires changes in the instruction set architecture.

e. Adding more pipeline stages always improves performance.

Short Answer

3. Given the following bit pattern (written in the same order as the book does, MSB first):

000000|10000|00001|00010|00000|100110
0 16 1 2 38 = xor

What does it represent on a MIPS computer, assuming that it is a MIPS instruction? (4 points)

xor $r2, $r16, $r1

-1 for wrong register order
-3 for opcode only
-2 for wrong opcode, correct registers

What does it represent on a MIPS computer, assuming that it is a 32-bit integer? (4 points)

0x02011026 = 33,624,102

$(6 + 16^1 \cdot 2 + 16^2 \cdot 0 + 16^3 \cdot 2 + 16^4 \cdot 1 + 16^5 \cdot 0 + 16^6 \cdot 2)$

-1 for not computing explicitly

4. Using only the XOR, AND, OR, and SLL instructions (and perhaps not even all of those), write the shortest possible MIPS program to add two unsigned numbers each of which is either zero or one (i.e. the sum will be zero, one, or two). Assume that the source values are in $r2 and $r3, and that the result is stored in $r4. (8 points)

```
AND $r4, $r2, $r3
SLL $r4, $r4, 1
XOR $r5, $r2, $r3
OR  $r4, $r4, $r5
```

-1 for forgetting to use temp
-1 for OR instead of XOR in instr 3

OR

```
AND $r4, $r2, $r3
SLL $r4, $r4, 1
XOR $r4, $r4, $r2
XOR $r4, $r4, $r3
```

5. Suppose that we have a computer with a four entry, direct-mapped cache. The block size is one word. We use the usual algorithm for mapping addresses to cache lines for a direct mapped cache. What addresses and data are stored in the cache after the following stream of addresses and data are read? For simplicity, assume that the full address is used as the cache tag, even though the two low-order bits would normally be omitted. (8 points)

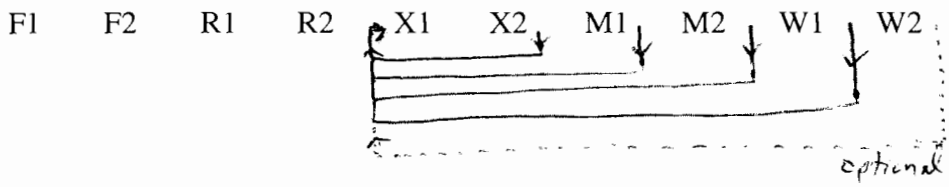
Address/Data stream:

block 0 ← Address=0, Data=23
block 1 ← Address=9, Data=120
block 0 ← Address = 8, Data = 32
block 2 ← Address = 6, Data = 49
block 1 ← Address = 9, Data = ~~82~~ 120
block 3 ← Address = 7, Data = 18
block 2 ← Address = 2, Data = 203

Answer:

block 0: address = 8 Data = 32
block 1: address = 9 Data = 120
block 2: address = 2 Data = 203
block 3: address = 7 Data = 18

6. The BevoTech computer company has designed a new 10-stage pipeline (shown below) by splitting each of the standard pipeline stages into two separate stages. The new pipeline is designed to perform better than the standard 5-stage pipeline. The first of each pair of stages does not produce any "useful" results for the purpose of forwarding – for example the values of operands read from the register file are not available until the end of the R2 stage, so the R1 stage does not produce any values that can be used elsewhere. Similarly, the ALU result is not available until the end of the X2 stage. (20 points)



a) If the new pipeline did not introduce any data or control hazards, and if latch overhead is negligible, how much faster would we expect the new machine to run compared to the standard 5-stage machine?

2 points

Twice as fast – the deeper pipeline should allow us to double the clock rate.

b) Is it possible to eliminate all read-after-write data hazards in this pipeline by using forwarding? Why or why not?

2 points

No. If instruction 'n' writes to a register and instruction 'n+1' reads from that same register, the result is not yet available to be forwarded. No result is ready after stage X1.

c) Describe the bypass paths needed to minimize stalls due to read-after-write data hazards. You can do this by listing the source and destination pipeline stages for each bypass path, or by just drawing the bypass paths on the pipeline diagram above.

6 points

see above

the last path might or might not be needed, depending on other details of the design

-3 points for just putting the two standard paths
-5 points for paths not to X1 (badly off)
-4 points for just one bypass path from X2

d) In the code fragments below, identify the data hazards and describe the number of pipeline stalls required even with your full bypassing.

```
mul r4, r5, r6
add r8, r9, r4
```

4 points
2 for hazard
2 for stall count

data RAW hazard
1 stall cycle

```
lw r1, 0(r2)
add r3, r1, r4
```

4 points
2 for hazard
2 for stall count

data RAW hazard
result available at end of M2, needed at start of X1
so 3 stall cycles

e) Suppose that branches are predicted as not-taken until the actual branch decision is known. Also, suppose that the branch target is known at the end of the X2 stage. How many in-progress instructions must be flushed when a branch is taken? How did you come up with this answer?

2 points

5 instructions must be flushed

1 cycle later, pipeline looks like:

F1	F2	R1	R2	X1	X2	M1	M2	W1	W2
Branch Target Instruction	(Flush)	(Flush)	(Flush)	(Flush)	(Flush)	Branch Instruction			

-0.5 for 6 instead of 5

Analytical Problems

7. The MIPS instruction set provides instructions which support conditional branching (beq, etc.). It is also possible to implement instructions which perform other operations conditionally. One such instruction is a conditional move. (12 points)

Suppose a new instruction is added to the MIPS instruction set:

```
cmove $r1, $r2, $r3
```

This instruction works as follows:

```
if (r3 == 0) then
  r1 = r2
else
  do nothing
```

In other words, the instruction only performs the move from r2 to r1 if r3 is zero.

Assume that this instruction is used to replace the following 2 instruction MIPS sequence wherever it appears:

```
        bne $r3, $r0, skip
        move $r1, $r2
skip:   ...
```

Suppose that 20% of the conditional branches in a program are used in this manner, so that the two instruction sequence can be replaced with a conditional move. Assume that the average CPI of the 'cmove' instruction is the same as the average CPI of the 'move' instruction. Thus, the effect of this change is to eliminate the time required by the branch instruction. What is the overall program speedup from this change, given the data in the following table?

Instruction Category	Frequency	Avg CPI
Arithmetic	43%	1.0
Data Transfer	40%	1.4
Conditional Branch	15%	1.8
Other	2%	1.3

speedup = Old time / New time = (.43(1) + .40(1.4) + .15(1.8) + .02(1.3)) / (.43(1) + .40(1.4) + .15[.8(1.8) + .2(0)] + .02(1.3))

(.43 + .56 + .27 + .026) / (.43 + .56 + .15[1.44] + .026) = 1.286 / 1.232 = 1.044 (1.04383)

.216

= 4.4% speedup

- 8 no CPI weighting — badly mis
- 6 points for not computing Branch+Mov → Branch correctly
- 2 points for minor error in Branch+Mov → Branch
- 2 points for simple speedup vs. time confusion

1.019 ⇒ (1.014)

-3 points for CPI vs. Speedup con

8. Write MIPS code, as short as possible, which accomplishes the same task as the following Java program. You may use the constant addresses "a" and "s" within your MIPS program. You must leave the result in memory location "s" when your MIPS code completes. Assume that "int" is a 32 bit signed integer. (15 points)

```
int a[10];
int s;

s = 0;
for (i=1; i<10; i++) {
  s = s + i*a[i];
}
```

```
        la   $r1, a      # address of 'a'
        la   $r2, s      # address of 's'
        li   $r3, 0      # s = 0
        li   $r4, 1      # i = 1
loop:   sll  $r5, $r4, 2 # i*4
        add  $r6, $r1, $r5  # compute address a[i]
        lw   $r7, ($r6)     # load a[i]
        mul  $r8, $r4, $r7  # i*a[i]
        add  $r3, $r3, $r8  # s=s+...
        addi $r4, $r4, 1    # i++
        addi $r9, $r4, -10  # compare...
        bltz $r9, loop      # ...
        sw   $r3, ($r2)     # store s
```

Can do even better than this, which got you extra credit of +2 in a few cases

close = 13 points (careless)

a few errors = 10 points

errors + use of s/a as operand = 7 points

9. The MIPS lui instruction loads a 16-bit immediate value into the upper 16 bits of a 32-bit register. The lower 16 bits of the register are set to zero. The MIPS single-cycle data path shown on the next page is not currently able to execute the lui instruction. (20 points)

10 pts

a) Add to the diagram on the next page the data paths, multiplexors, control lines, etc. required to support the lui instruction in the simplest manner possible.

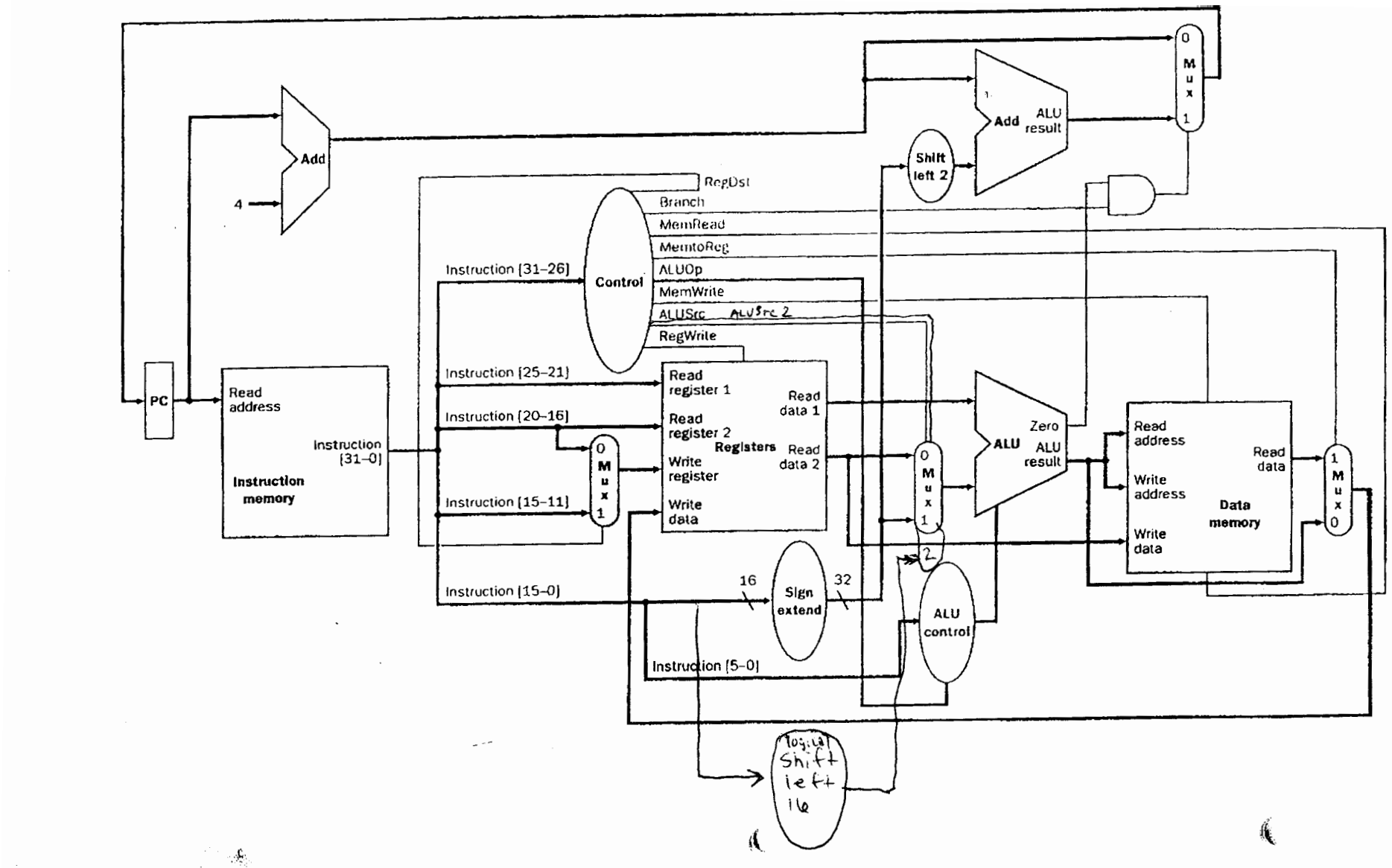
10 pts

b) Specify the values of the control signals (0, 1, or X for don't care) when executing an lui instruction. If you added any new control signals in part (a), put those at the bottom of the table.

Points

	SIGNAL	VALUE
3p	RegDst	0
1	Branch	0
	MemRead	0 or X
1	MemtoReg	0
	ALUOp	(see note)
1	MemWrite	0
	ALUSrc	0
3 1	RegWrite	1
	ALUSrc2	1

Note: You do not have to provide a value for the ALUOp control signal. You can assume that it is set such that the ALU will feed its bottom input through to its output, without changing the value.


Add
4
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [31–26]
Control
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
0 M u x 1
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign extend
32
Shift left 2
Add
ALU result
ALU
Zero
ALU control
2
Read address
Write address
Write data
Data memory
Read data
1 M u x 0
Logical Shift left 16