Computer Science 605.611

Mid Term Exam
Brian Loughran

1. (4) Write down a sequence of one or more MIPS instructions (only built-in true-ops with valid addressing modes and no pseudo-instructions) to subtract the two's complement signed integer in a single 8-bit memory byte from the contents of register $10 and put the result back into $10. The address of the memory byte is 0x403C0017.

```
lui $1,0x403C
lb $1,0x17($1)          # load contents of 0x403C0017 to $1
sub $10,$10,$1          # perform subtraction
```

2. A tenth instruction is to be included in our MIPS 9-instruction core subset (add, sub, and, or, slt, lw, sw, beq, j). This new true-op instruction is addm rt,disp(rs) which computes the sum of the contents of a memory word plus the contents of the rt register and places the sum back into the rt register.

a) (3) Show the MIPS machine code format required for this new true-op machine instruction. That is, show which bits within the 32-bit machine instruction correspond to each part of the instruction.

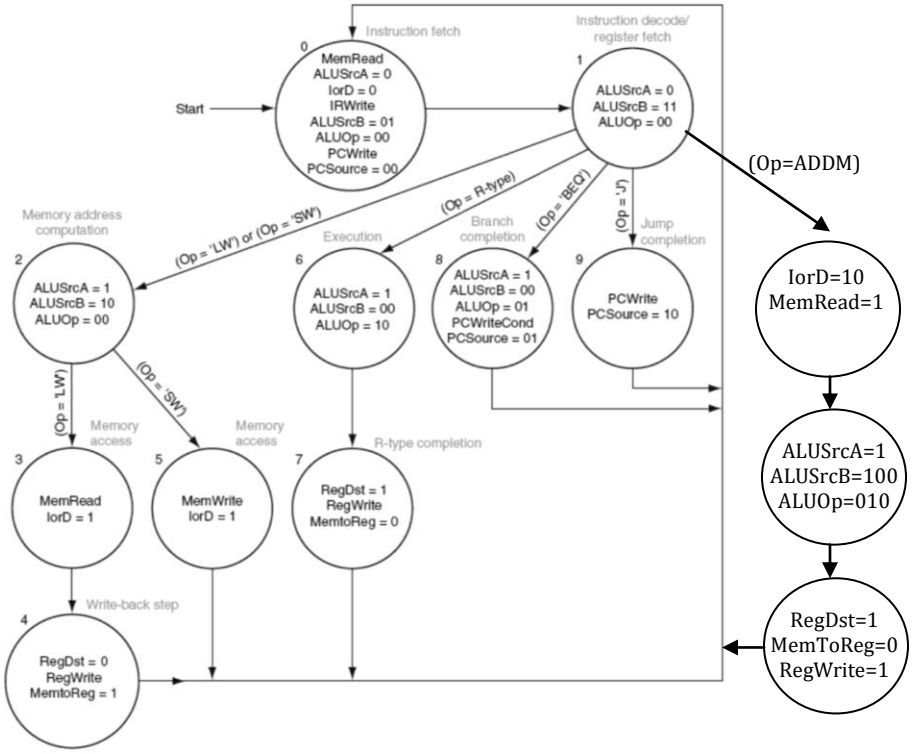
This instruction most closely resembles an I-type instruction, so we will use a format similar:

6 bits (0-5)	5 bits (6-10)	5 bits (11-15)	16 bits (16-31)
opcode	rs	rt	disp

b) (8) Modify the FSM (finite state machine) diagram shown below to support this new instruction on the multi-cycle datapath. Show any new states required and the control bits (existing or new) that affect each new state as well as the transitions into and out of each new state.

We will have to add a new bit to the lower B input to the ALU, as there would now be 5 possible candidates, the current 4 and the addm instruction. ALUSrcB would be expanded from 2 bits to 3 bits.

We will add three new states to the FSM to support the new instruction on the multi-cycle datapath. The first state we add, which branches off of state 1 from the original FSM, we read the memory based on the given address. In the second state we add, we compute the sum of the given register and the memory value. In the third state we add we write the result to the given register. See diagram below:


Instruction fetch
Instruction decode/
register fetch
0
MemRead
ALUSrcA = 0
IorD = 0
IRWrite
ALUSrcB = 01
ALUOp = 00
PCWrite
PCSource = 00
Start
1
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00
(Op=ADDM)
(Op = 'LW') or (Op = 'SW')
(Op = R-type)
(Op = 'BEQ')
(Op = 'J')
Memory address
computation
Execution
Branch
completion
Jump
completion
2
ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00
6
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 10
8
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWriteCond
PCSource = 01
9
PCWrite
PCSource = 10
IorD=10
MemRead=1
(Op = 'LW')
(Op = 'SW')
Memory
access
Memory
access
R-type completion
3
MemRead
IorD = 1
5
MemWrite
IorD = 1
7
RegDst = 1
RegWrite
MemtoReg = 0
ALUSrcA=1
ALUSrcB=100
ALUOp=010
Write-back step
4
RegDst = 0
RegWrite
MemtoReg = 1
RegDst=1
MemToReg=0
RegWrite=1

c) (3) Are changes required to the multi-cycle datapath to support this new true-op instruction? If so, describe the required changes.

The multi-cycle datapath requires two small changes. First we connect the memory address to the memory unit so we can read from that address. Second we connect the memory data register in to the ALU for addition with rs.

3. (5) The instruction ror (rotate right) is a pseudo-instruction. It shifts the bits within a register right the indicated number of positions. The bits shifted out on the right are wrapped around and enter the left end of the register in a circular fashion. For example, if register $14 contains 0x12345678 then ror $8,$14,16 rotates register $14 by 16 bits to produce 0x56781234 in register $8.
Show a series of at most 3 true-op instructions that place into register $8 the same result as that produced by the instruction ror $8,$14,10 which rotates the contents of register $14 right by 10 bits and places the rotated bit pattern into register $8 (the solution should work for any original value in $14 and should use no pseudo-instructions).

```
srl $1,$14,10
sll $2,$14,26
or $8,$1,$2
```

4. Suppose that VAL1 is the name of a memory variable to which the memory address 0x10048840 is assigned. The variable currently contains the value -3. Use eight hex digits to show the machine code for each of the true-ops that are executed for each of the assembly language instructions (pseudo-instruction or true-op) listed below. One or more true-ops may be required for each assembly language instruction.

a) (3) la $8,VAL1

la $8,val1 =

```
lui $at,0x1004          # I-type, op: 15, rt: 1, immed: 0x1004
ori $8,$at,0x8840       # I-type, op: 13, rs: 1, rt: 8, immed: 0x8840

# hex code
0x3c011004
0x34288840
```

b) (3) lw $4, 0($8)

true op: I-type, op: 35, rs: 8, rt: 4, immed: 0x0000

hex code

0x8d040000

c) (2) addiu $6,$6,0x10048840 =

```
lui $1,0x1004          # I-type, op: 15, rt: 1, immed: 0x1004
lw $1,0x8840($1)       # I-type, op: 35, rs: 1, rt: 1, immed: 0x8840
addu $6,$6,$1          # R-type, op: 0, rs: 6, rt: 1, rd: 6
```

```
# hex code
0x3c011004
0x8c218840
0x00c13021
```

5. (4) List the name and value (0 for deasserted or don't care and 1 for asserted) of each of the control bits used in the single-cycle datapath when executing the machine instruction that corresponds to: sw $4,8($2).

Bit	RegDst	ALU Src	Memt oReg	RegWrite	Mem Read	Mem Write	Branch	AluOp1	ALUOp0	Jump
val	0	1	0	0	0	1	0	0	0	0

6. a) (4) The sign extension unit within the MIPS datapath expands the rightmost 16 bits in I-type machine instructions into the equivalent 32-bit signed integer. Write down a pair (i.e., two) MIPS true-op instructions that place into register $9 the 32-bit two's complement equivalent of the two's complement integer contained in the low (i.e., rightmost) 14 bits of $9. The solution should work for any original value in $9.

```
sll $9, $9 18          # isolate rightmost 14 bits
sra $9, $9, 18         # sra shifts in the sign bit, maintain value of lower 14 bits
```

b) (4) Since the MIPS CPU registers are all 32-bit registers, an even/odd register pair is used to hold a single 64-bit integer. Assume that $2 contains the high 32 bits and $3 contains the low 32 bits of a 64-bit two's complement integer. Also assume that $6 contains the high 32 bits and $7 contains the low 32 bits of another two's complement integer. Write down a sequence of at most 4 MIPS true-op instructions to place the sum of these two 64-bit integers into registers $2 and $3.

```
add $3,$7,$3           # add the least significant bits
slt $1,$3,$7           # check if overflow ($1=1 if overflow; else $1=0)
add $2,$6,$2           # add the most significant bits
add $1,$2,$2           # add carry bit
```

7. Along with the PC (program counter) register, our MIPS processor contains 32 general purpose registers ($0 through $31). Register $0 is hardwired to zero and can't be modified. Assume that each register contains an integer = 4 times the register number. That is, register $n contains the value 4*n, so $1 contains 4, $2 contains 8, $3 contains 12, etc. A MIPS machine instruction that resides at memory address 0x4000C408 is executed. Indicate all registers and/or memory locations that are modified along with the new value produced by executing this machine instruction if the instruction is:

a) (5) 0x04C0FFFE
=0000 0100 1100 0000 1111 1111 1111 1110
bltz, $6 0xFFFE
register 6 is not less than 0
memory incremented 4 to proceed to next instruction at 0x4000C40C

b) (5) 0x04C10002
=0000 0100 1100 0001 0000 0000 0000 0010
bgez $6, 0x0002
register 6 is greater than 0
memory is incremented by 0x0002*4=0x0008
new memory address is 0x4000C410

8. (8) A program used by a financial institution is to add the dollar amount $100.45 to a current account balance of $109,776,749.55 . Floating point register $f4 contains the IEEE 754 representation of the current balance ($109,776,749.55) and register $f6 contains the IEEE 754 representation of the amount to be added ($100.45). What dollar amount is represented by the IEEE 754 pattern produced in register $f8 by the instruction add.s $f8,$f4,$f6 if no round or guard bits are used? Register $f8 contains the representation of :

$109,776,749.55 = 0 10011001 10100010110000111101110
$100.45 = 0 10000101 10010001110011001100110
Shifting bits: 0 10011001 00000000000000000000100
Adding signicicands: 0 10011001 10100010010000111110010
No overflow
Convert to float: $109,776,784.00
Some value lost to rounding in this case

9. Consider the following instruction sequence:

```
lui  $4,1
srl  $5,$4,16
sll  $4,$4,15
```

```
and   $4,$6,$4
sra   $4,$4,31
xor   $6,$6,$4
addu  $6,$6,$5
```

a) (5) How many clock cycles are required to execute this instruction sequence on our 5-stage MIPS pipeline with a data hazard unit but no forwarding unit?

Cycle #	Fetch	Reg Read	ALU Op	Mem Access	Reg Write
1	lui $4,1				
2	srl $5,$4,16	lui $4,1			
3	sll $4,$4,15	srl $5,$4,16	lui $4,1		
4	sll $4,$4,15	srl $5,$4,16		lui $4,1	
5	sll $4,$4,15	srl $5,$4,16			lui $4,1
6	and $4,$6,$4	sll $4,$4,15	srl $5,$4,16		
7	sra $4,$4,31	and $4,$6,$4	sll $4,$4,15	srl $5,$4,16	
8	sra $4,$4,31	and $4,$6,$4		sll $4,$4,15	srl $5,$4,16
9	sra $4,$4,31	and $4,$6,$4			sll $4,$4,15
10	xor $6,$6,$4	sra $4,$4,31	and $4,$6,$4		
11	xor $6,$6,$4	sra $4,$4,31		and $4,$6,$4	
12	xor $6,$6,$4	sra $4,$4,31			and $4,$6,$4
13	addu $6,$6,$5	xor $6,$6,$4	sra $4,$4,31		
14	addu $6,$6,$5	xor $6,$6,$4		sra $4,$4,31	
15	addu $6,$6,$5	xor $6,$6,$4			sra $4,$4,31
16		addu $6,$6,$5	xor $6,$6,$4		
17		addu $6,$6,$5		xor $6,$6,$4	
18		addu $6,$6,$5			xor $6,$6,$4
19			addu $6,$6,$5		
20				addu $6,$6,$5	
21					addu $6,$6,$5

21 cycles

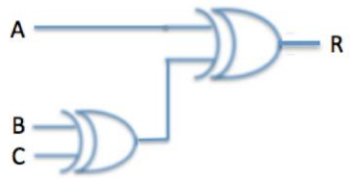
b) (5) How many clock cycles are required to execute this instruction sequence on our 5-stage MIPS pipeline with both a data hazard unit and a forwarding unit?

7 instructions with a forwarding unit will require 11 cycles (7 instructions plus 4 for the first lui to advance to the RegWrite cycle)

c) (5) Write down a single MIPS assembly language instruction (it can be a true-op or pseudo-instruction) that has the same effect on register $6 as the above sequence of six instructions.

addi $6,$6,1

10. (5) The combination of logic gates shown below implements one or more of the functions listed. Identify all functions [a), b) c), d)] that are correctly implemented by this combination of logic gates.



a) majority function (i.e., R=1 if at least 2 of the inputs are 1)
b) arithmetic sum (i.e., R = the arithmetic sum of the 3 inputs)
c) even parity (i.e., R=1 if an even number of the inputs = 1)
d) odd parity (i.e., R=1 if an odd number of the inputs = 1)

truth table:

case	a	b	c	b XOR c	a XOR (b XOR c)
1	0	0	0	0	0
2	0	0	1	1	1
3	0	1	0	1	1
4	1	0	0	0	1
5	0	1	1	0	0
6	1	0	1	1	0
7	1	1	0	1	0
8	1	1	1	0	1

a) false -> see case 2, 3, 4, 5, 6, 7
b) true -> see case 1-8 (assuming the sum of 1 and 1 is 0)
c) false -> see case 1-8
d) true -> see case 1-8

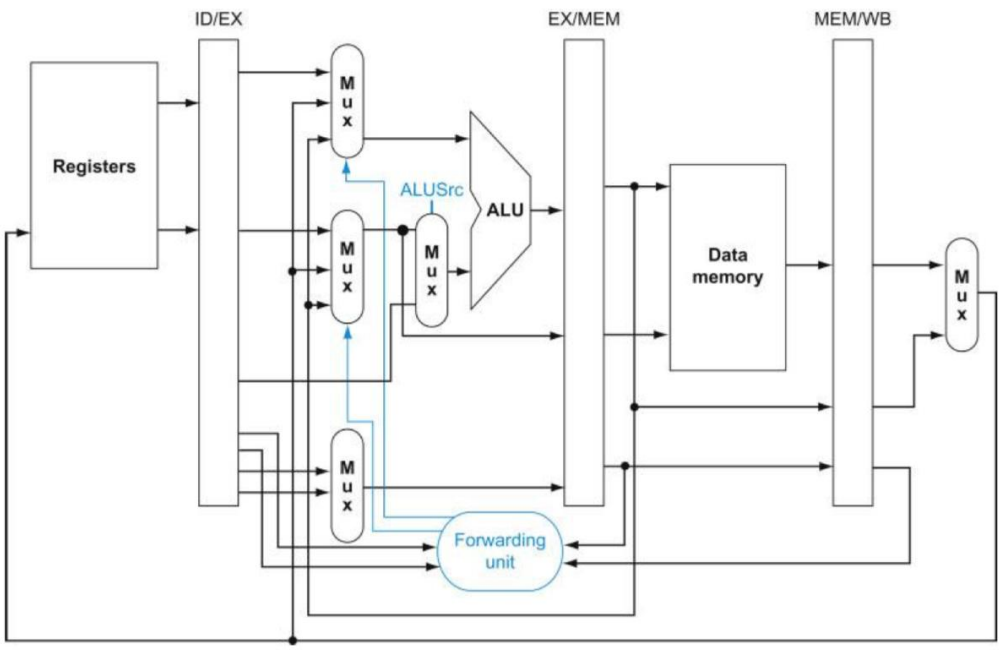
This is an example of odd parity and arithmetic sum (assuming the sum of 1 and 1 is 0)

11. Consider the instruction sequence shown below:

```
xor   $2,$0,$2
lw    $9,16($8)
srl   $2,$2,1
add   $3,$2,$2
or    $5,$0,$9
sw    $3,16($8)
sub   $4,$0,$3
```

a) (4) What is the native MIPS rating for this instruction sequence if it is executed on our 5-stage MIPS pipeline with a clock cycle time of 400 ns and with both a data

hazard unit and a forwarding unit? The forwarding unit is located in the execute stage as shown below:



Cycle	IF	ID	EX	MEM	WB
1	xor $2,$0,$2				
2	lw $9,16($8)	xor $2,$0,$2			
3	srl $2,$2,1	lw $9,16($8)	xor $2,$0,$2		
4	add $3,$2,$2	srl $2,$2,1	lw $9,16($8)	xor $2,$0,$2	
5	or $5,$0,$9	add $3,$2,$2	srl $2,$2,1	lw $9,16($8)	xor $2,$0,$2
6	sw $3,16($8)	or $5,$0,$9	add $3,$2,$2	srl $2,$2,1	lw $9,16($8)
7	sub $4,$0,$3	sw $3,16($8)	or $5,$0,$9	add $3,$2,$2	srl $2,$2,1
8		sub $4,$0,$3	sw $3,16($8)	or $5,$0,$9	add $3,$2,$2
9		sub $4,$0,$3		sw $3,16($8)	or $5,$0,$9
10			sub $4,$0,$3		sw $3,16($8)
11				sub $4,$0,$3	
12					sub $4,$0,$3

Therefore 7 instructions take 12 cycles, thus
CPI = 12/7
Clock rate = 1 / clock cycle time = 1 / (400 * 10 ^ -9) = 2.5 * 10 ^ 6
Native MIPS = clock rate / (10^6 * CPI)
= 2.5 * 10 ^ 6 / ((12/7) * 10^6)
Native MIPS rating = 1.458

b) (3) Show the proper settings for the following control bits when the instruction sw $3,16($8) is in the execute stage:

ForwardA = 00
ForwardB = 00
ALUSrc = 1

c) (3) What is the native MIPS rating for this instruction sequence if it is executed on our 5-stage MIPS pipeline with a clock cycle time of 400 ns and only a data hazard unit (no forwarding unit)?

Cycle	IF	ID	EX	MEM	WB
1	xor $2,$0,$2				
2	lw $9,16($8)	xor $2,$0,$2			
3	srl $2,$2,1	lw $9,16($8)	xor $2,$0,$2		
4	add $3,$2,$2	srl $2,$2,1	lw $9,16($8)	xor $2,$0,$2	
5	add $3,$2,$2	srl $2,$2,1		lw $9,16($8)	xor $2,$0,$2
6	or $5,$0,$9	add $3,$2,$2	srl $2,$2,1		lw $9,16($8)
7	or $5,$0,$9	add $3,$2,$2		srl $2,$2,1	
8	or $5,$0,$9	add $3,$2,$2			srl $2,$2,1
9	sw $3,16($8)	or $5,$0,$9	add $3,$2,$2		
10	sub $4,$0,$3	sw $3,16($8)	or $5,$0,$9	add $3,$2,$2	
11		sub $4,$0,$3	sw $3,16($8)	or $5,$0,$9	add $3,$2,$2
12		sub $4,$0,$3		sw $3,16($8)	or $5,$0,$9
13		sub $4,$0,$3			sw $3,16($8)
14			sub $4,$0,$3		
15				sub $4,$0,$3	
16					sub $4,$0,$3

Therefore 7 instructions take 16 cycles, thus
CPI = 16/7
Native MIPS = clock rate / (10^6 * CPI)
= 2.5 * 10 ^ 6 / ((16/7) * 10^6)
Native MIPS rating = 1.094

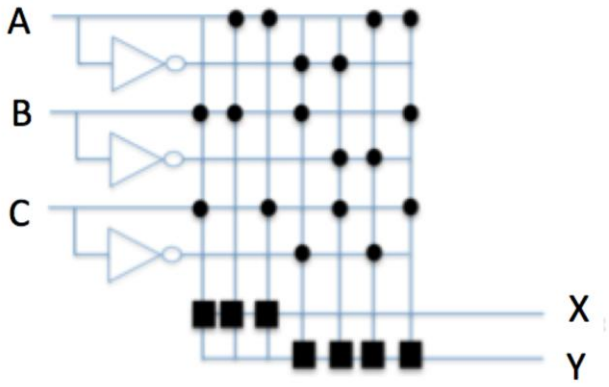
d) (3) What is the native MIPS rating for this instruction sequence if it is executed on the multi-cycle datapath with a clock cycle time of 400 ns?

xor	$2,$0,$2	R-type	4 cycles
lw	$9,16($8)	I-type	5 cycles
srl	$2,$2,1	R-type	4 cycles
add	$3,$2,$2	R-type	4 cycles
or	$5,$0,$9	R-type	4 cycles
sw	$3,16($8)	I-type	4 cycles
sub	$4,$0,$3	R-type	4 cycles

Therefore 7 instructions take 29 cycles, thus
CPI = 29/7

Native MIPS = clock rate / (10^6 * CPI)
= 2.5 * 10 ^ 6 / ((29/7) * 10^6)
Native MIPS rating = 0.603

12. In the PLA shown below, AND gate plane connections are denoted by a dot • while OR gate plane connections are denoted by a square ■ . The PLA shows how each output (X and Y) is related to the inputs (A, B, and C). Other ways to show how the outputs are related to the inputs include truth tables and logical expressions or functions.



(6) Fill in the columns for the outputs X and Y in the following truth table so that the table corresponds to this PLA.

A	B	C	X	Y
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1