ECE 2030 B 12:00pm	Computer Engineering	Fall 2010
4 problems, 7 pages	Exam Three	1 December 2010

Instructions: This is a closed book, closed note exam. Calculators are not permitted. If you have a question, raise your hand and I will come to you. Please work the exam in pencil and do not separate the pages of the exam. For maximum credit, show your work.
Good Luck!

Your Name (***please print***) ____

1	2	3	4	total
18	32	25	25	100

Problem 1 (2 parts, 18 points) Datapath Elements

Part A (9 points) Consider the following input and output values for a shift operation. Determine the shift *type* and *amount* required to achieve the listed transformation. I/Os are in hexadecimal.

Input Value	Output Value	Shift Type	Shift Amount (signed decimal value)
87654321	FFF87654		
87654321	00000008		
87654321	43218765		

Part B (9 points) Consider the following input and output values for a logical operation. Determine the *logical function* and *function code* (in hexadecimal) required for the operation.

X Input	Y Input	Output	Logical Function	Function Code
87654321	00FF00FF	00650021		
87654321	00FF00FF	879A43DE		
87654321	00FF00FF	789ABCDE		

Problem 2 (3 parts, 32 points) Memory Systems

Part A (12 points) Consider a DRAM chip organized as **64 million addresses** of **16 bit words**. Assume both the DRAM cell and the DRAM chip is square. The column number and offset concatenate to form the memory address. Using the organization approach discussed in class, answer the following questions about the chip. **Express all answers in decimal**.

number of columns ____________

column decoder required (*n* to *m*) ____________

type of mux required (*n* to *m*) ____________

number of muxes required ____________

number of address lines in column number ____________

number of address lines in column offset ____________

Part B (10 points) Consider a **4 Gbyte** memory system with **512 million addresses** of **8 byte words** using DRAM chips organized as **64 million** addresses by **16 bit words.**

word address lines for memory system ____________

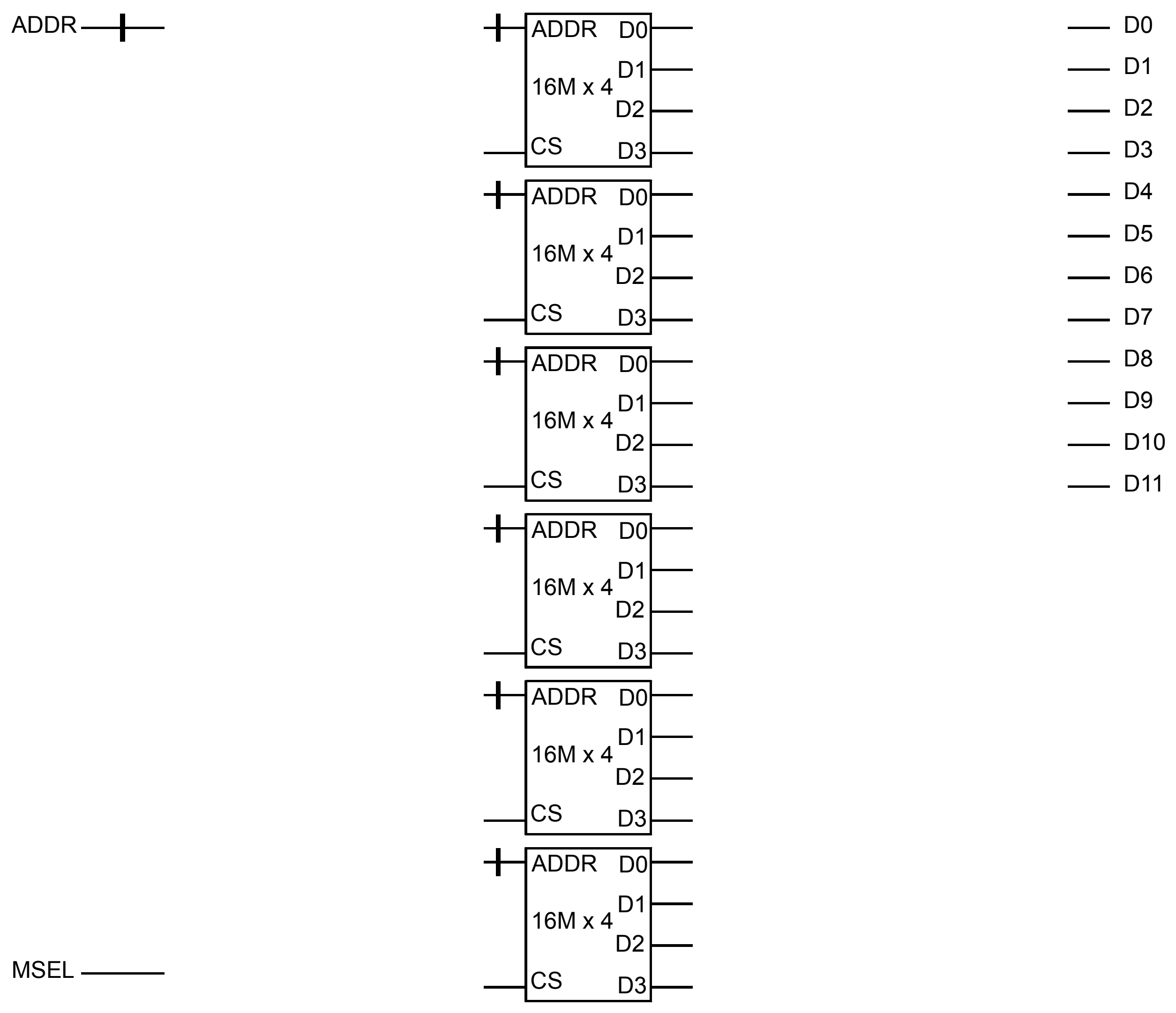
chips needed in one bank ____________

banks for memory system ____________

memory decoder required (*n* to *m*) ____________

DRAM chips required ____________

Part C (10 points) Design a 32 M address x 12 bit memory system with six 16 M address x 4 bit memory chips. **Label all busses and indicate bit width**. Assume R/W is connected and not shown here. Use a decoder if necessary. Place a star on the chip(s) that contain address 26,000,000.



Problem 3 (5 parts, 25 points) Microcode

Using the supplied datapath, write microcode fragments to accomplish the following procedures. Express all values in hexadecimal notation. Use 'X' when a value is don't cared. Smile when you're happy. For maximum credit, complete the description field.

Part A (6 points) \$6 ← mem[0x3000]. **Use only register 6.**

#	*X*	*Y*	*Z*	*rwe*	*im en*	*im va*	*au en*	*-a /s*	*lu en*	*lf*	*su en*	*st*	*ld en*	*st en*	*r/ -w*	*msel*	*description*
1																	
2																	

Part B (5 points) Mask all but the eight least significant bits of \$8 **Use only register 8.**

#	*X*	*Y*	*Z*	*rwe*	*im en*	*im va*	*au en*	*-a /s*	*lu en*	*lf*	*su en*	*st*	*ld en*	*st en*	*r/ -w*	*msel*	*description*
1																	

Part C (4 points) Subtract 8 from register \$3. **Use only registers 3.**

#	*X*	*Y*	*Z*	*rwe*	*im en*	*im va*	*au en*	*-a /s*	*lu en*	*lf*	*su en*	*st*	*ld en*	*st en*	*r/ -w*	*msel*	*description*
1																	

Part D (6 points) mem[\$4] ← 0x1957. **Use only registers 4 and 5.**

#	*X*	*Y*	*Z*	*rwe*	*im en*	*im va*	*au en*	*-a /s*	*lu en*	*lf*	*su en*	*st*	*ld en*	*st en*	*r/ -w*	*msel*	*description*
1																	
2																	

Part E (4 points) \$7 ← 32 * \$7. **Use only register 7.**

#	*X*	*Y*	*Z*	*rwe*	*im en*	*im va*	*au en*	*-a /s*	*lu en*	*lf*	*su en*	*st*	*ld en*	*st en*	*r/ -w*	*msel*	*description*
1																	

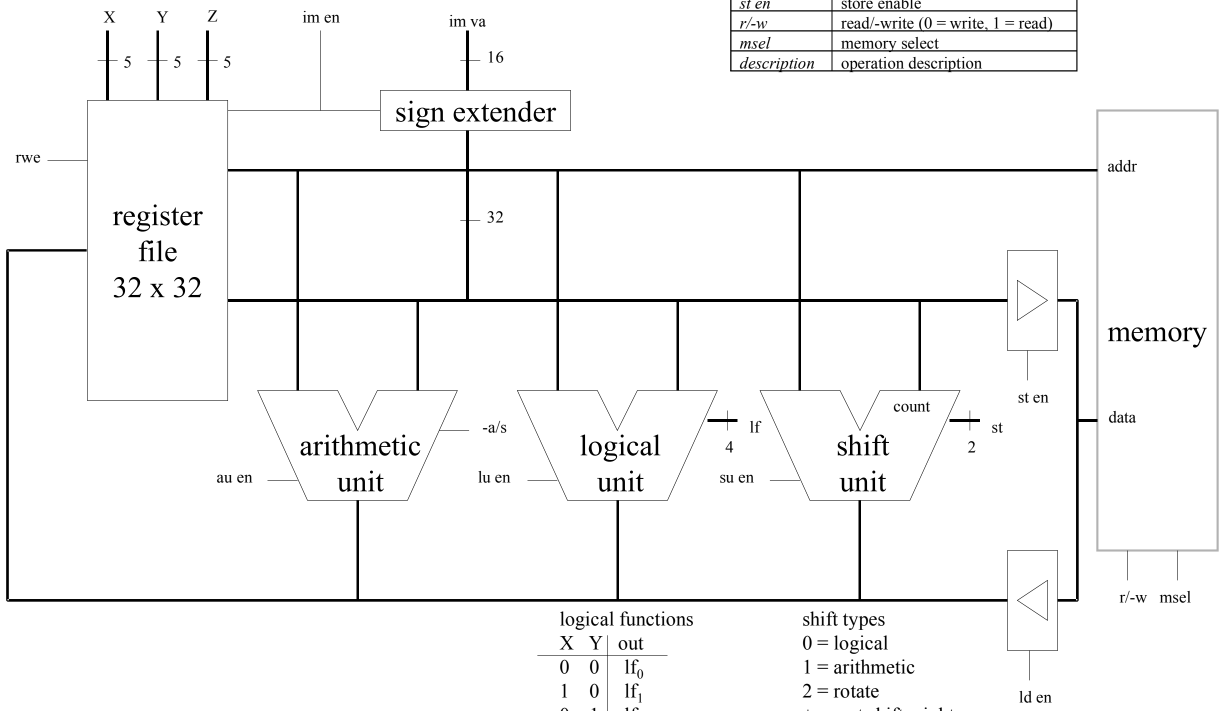
Problem 4 (1 part, 25 points) Assembly Programming

Part A (25 points) Complete this subroutine that averages an array of integers in memory. Assume $4 contains the number of integers and $5 contains the starting address of the array. Return the average in $7. Use the $8 for the input value, $7 for the sum, and $6 for the end test.

label	*instruction*	*comment*
AvgN:		**# offset = 4 * # elems**
		# end = base + offset
		# load 1st element
		# point to 2nd element
		# load next element
		# add to sum
		# point to next element
		# loop until end
		# compute average
		# put average in $7
	jr $31	**# return to caller**

cycle	cycle number
X	register driven onto X bus
Y	register driven onto Y bus
Z	register written from Z bus
rwe	register write enable
im en	immediate enable on Y bus
im va	immediate value

au en	arithmetic unit enable
-a/s	-add / sub (0 = add, 1 = subtract)
lu en	logical unit enable
lf	logical function
su en	shift unit enable
st	shift type
ld en	load enable
st en	store enable
r/-w	read/-write (0 = write, 1 = read)
msel	memory select
description	operation description



logical functions

X	Y	out
0	0	lf_0
1	0	lf_1
0	1	lf_2
1	1	lf_3

shift types
0 = logical
1 = arithmetic
2 = rotate
+ count shifts right
- count shifts left

MIPS Instruction Set

instruction	***example***	***meaning***
arithmetic		
add	add $1,$2,$3	$1 = $2 + $3
subtract	sub $1,$2,$3	$1 = $2 - $3
add immediate	addi $1,$2,100	$1 = $2 + 100
add unsigned	addu $1,$2,$3	$1 = $2 + $3
subtract unsigned	subu $1,$2,$3	$1 = $2 - $3
add immediate unsigned	addiu $1,$2,100	$1 = $2 + 100
set if less than	slt $1, $2, $3	if ($2 < $3), $1 = 1 else $1 = 0
set if less than immediate	slti $1, $2, 100	if ($2 < 100), $1 = 1 else $1 = 0
set if less than unsigned	sltu $1, $2, $3	if ($2 < $3), $1 = 1 else $1 = 0
set if < immediate unsigned	sltui $1, $2, 100	if ($2 < 100), $1 = 1 else $1 = 0
multiply	mult $2,$3	Hi, Lo = $2 * $3, 64-bit signed product
multiply unsigned	multu $2,$3	Hi, Lo = $2 * $3, 64-bit unsigned product
divide	div $2,$3	Lo = $2 / $3, Hi = $2 mod $3
divide unsigned	divu $2,$3	Lo = $2 / $3, Hi = $2 mod $3, unsigned
transfer		
move from Hi	mfhi $1	$1 = Hi
move from Lo	mflo $1	$1 = Lo
load upper immediate	lui $1,100	$1 = 100 x 2^{16}
logic		
and	and $1,$2,$3	$1 = $2 & $3
or	or $1,$2,$3	$1 = $2 | $3
and immediate	andi $1,$2,100	$1 = $2 & 100
or immediate	ori $1,$2,100	$1 = $2 | 100
nor	nor $1,$2,$3	$1 = not($2 | $3)
xor	xor $1, $2, $3	$1 = $2 ⊕ $3
xor immediate	xori $1, $2, 255	$1 = $2 ⊕ 255
shift		
shift left logical	sll $1,$2,5	$1 = $2 << 5 (logical)
shift left logical variable	sllv $1,$2,$3	$1 = $2 << $3 (logical), variable shift amt
shift right logical	srl $1,$2,5	$1 = $2 >> 5 (logical)
shift right logical variable	srlv $1,$2,$3	$1 = $2 >> $3 (logical), variable shift amt
shift right arithmetic	sra $1,$2,5	$1 = $2 >> 5 (arithmetic)
shift right arithmetic variable	srav $1,$2,$3	$1 = $2 >> $3 (arithmetic), variable shift amt
memory		
load word	lw $1, 1000($2)	$1 = memory [$2+1000]
store word	sw $1, 1000($2)	memory [$2+1000] = $1
load byte	lb $1, 1002($2)	$1 = memory[$2+1002] in least sig. byte
load byte unsigned	lbu $1, 1002($2)	$1 = memory[$2+1002] in least sig. byte
store byte	sb $1, 1002($2)	memory[$2+1002] = $1 (byte modified only)
branch		
branch if equal	beq $1,$2,100	if ($1 = $2), PC = PC + 4 + (100*4)
branch if not equal	bne $1,$2,100	if ($1 ≠ $2), PC = PC + 4 + (100*4)
jump		
jump	j 10000	PC = 10000*4
jump register	jr $31	PC = $31
jump and link	jal 10000	$31 = PC + 4; PC = 10000*4