Final Exam

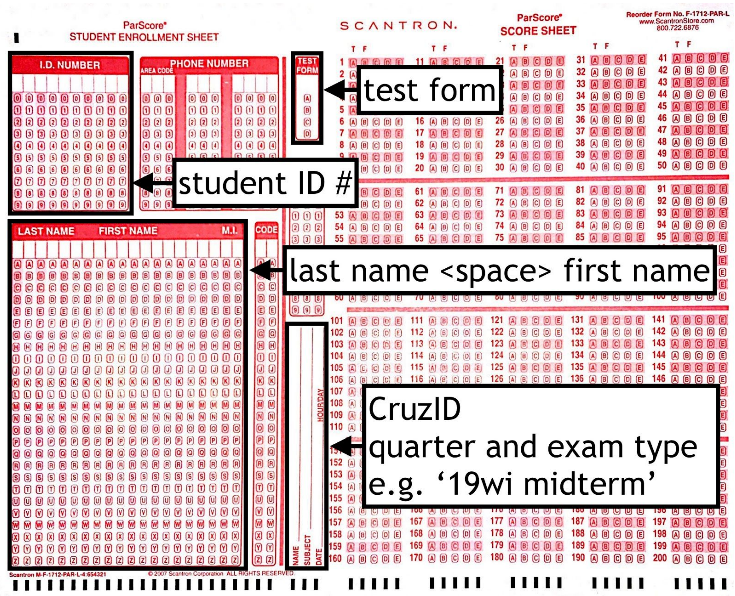
CMPE 012: Computer Systems and Assembly Language

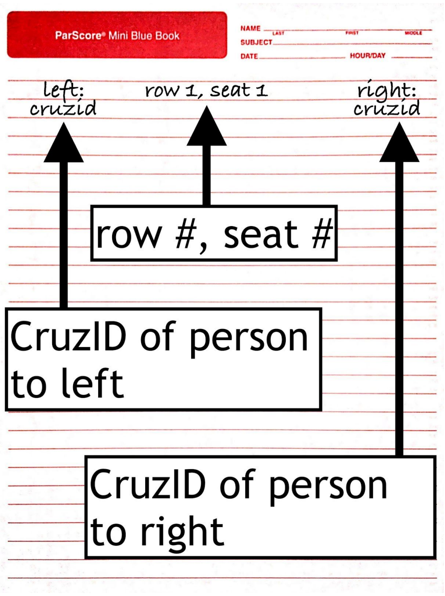
University of California, Santa Cruz

DO NOT BEGIN UNTIL YOU ARE TOLD TO DO SO.

This exam is closed book and closed notes. Only 4-function calculators are permitted. Answers must be marked on the Scantron form to be graded. All work must be written on the exam.

On the Scantron form, bubble in your name, student ID number, and test form (found in the footer of subsequent pages). In the center of the page write your CruzID, quarter, and exam type. On the back of the page, write the CruzIDs of students sitting to your left and right, and your row and seat number. See below.





On this page, write your last name, first name, CruzID, row and seat numbers, and the CruzIDs of the people to your immediate left and right. Once you are permitted to begin, write your CruzID on all subsequent pages of the exam.

You must sit in your assigned seat. Keep your student or government issued ID on your desk. Brimmed hats must be removed or turned around backwards. Only unmarked water bottles are permitted. Backpacks must be placed at the front of the room or along the walls. Your cell phone must be on a setting where it will not make noise or vibrate.

There are 42 questions on this exam; you only need to answer 40 for full points. The additional two questions (of your choosing) will be counted as extra credit. All questions are multiple choice, and some questions have more than one correct answer. **You must mark all correct answers to receive credit for a question.** Some true/false questions might list False as answer A and True as answer B. Follow the answers on the exam, **NOT** the T F notation on the Scantron Form. You will have 120 minutes to complete this exam.

Row #   Seat #   CruzID

Your Last Name   Your First Name

CruzID of person to left   CruzID of person to right

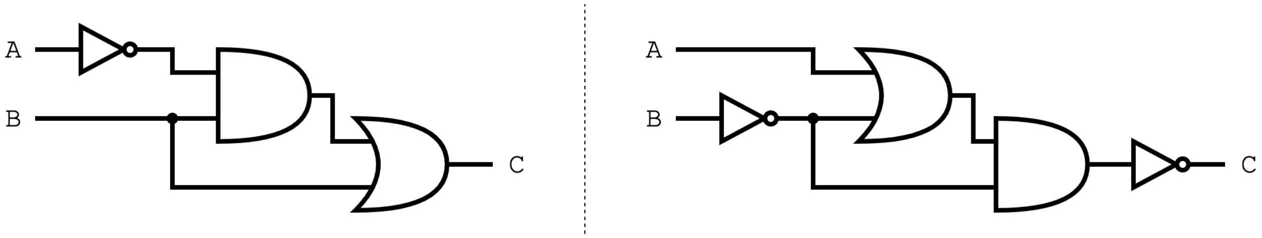
CruzID: ______________________ @ucsc.edu

CMPE 12 Final - Version A

Spring 2019

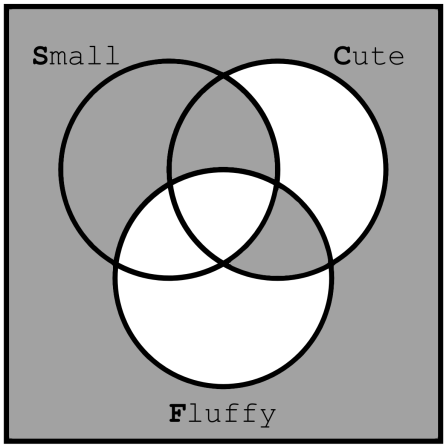
Combinational Logic & Boolean Algebra

1. True or False: These two circuits are logically equivalent.



○ A. True
○ B. False

2. Select the Boolean expression(s) matching the grey filled areas of this Venn diagram.

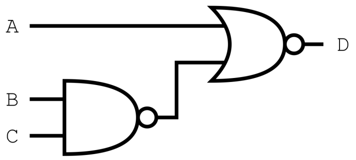


○ A. $SCF + \bar{S}C\bar{F} + S\bar{C}F + \bar{S}\bar{C}F$
○ B. $SCF + \bar{C}F + \bar{S}C\bar{F}$
○ C. $\bar{S}\bar{C}\bar{F} + S\bar{F} + \bar{S}FC$
○ D. Correct answer not listed
○ E. $\bar{S}\bar{C}\bar{F} + \bar{S}F + S\bar{F}C + CF$

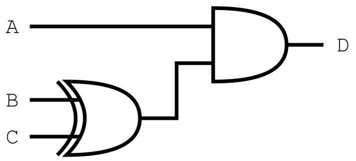
3. Which circuit matches this truth table?

A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

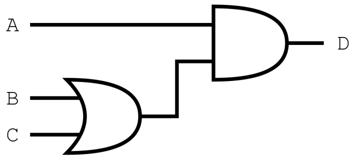
○ A.



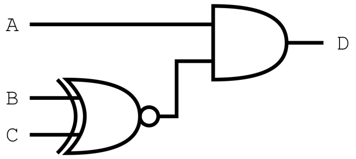
○ B.



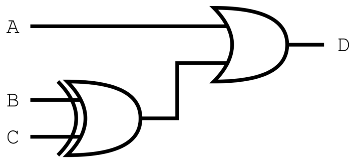
○ C.



○ D.

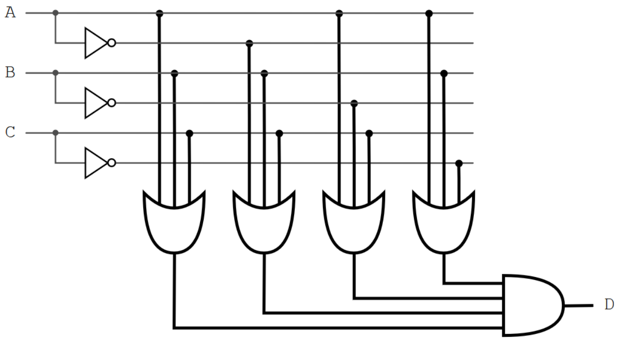


○ E.



4. What kind of multiplexor has 3 select lines?
   - ○ A. 3-to-1
   - ○ B. 2-to-1
   - ○ C. 16-to-1
   - ○ D. 8-to-1
   - ○ E. 9-to-1

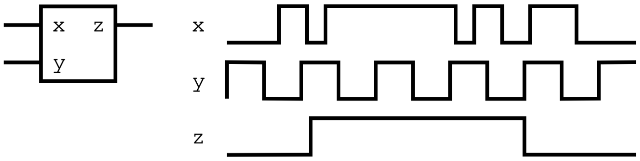
5. What equation does this PLA represent?



   - ○ A. $(\bar{A}+B+C)(A+\bar{B}+\bar{C})(A+B+C)(\bar{A}+\bar{B}+\bar{C})$
   - ○ B. $(\bar{A}+\bar{B}+\bar{C})(A+B+\bar{C})(\bar{A}+B+\bar{C})(\bar{A}+\bar{B}+C)$
   - ○ C. $(\bar{A}+\bar{B}+C)(\bar{A}+B+\bar{C})(A+\bar{B}+\bar{C})(A+\bar{B}+C)$
   - ○ D. $(A+B+C)(A+\bar{B}+\bar{C})(\bar{A}+B+\bar{C})(\bar{A}+\bar{B}+C)$
   - ○ E. $(A+B+C)(\bar{A}+B+C)(A+\bar{B}+C)(A+B+\bar{C})$

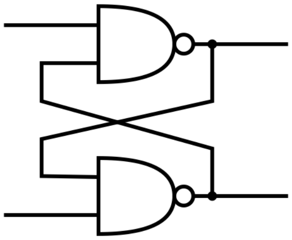
Sequential Logic

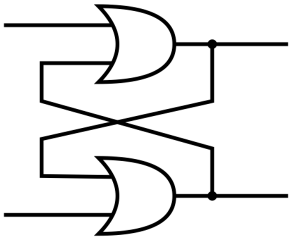
6. What device does this timing diagram represent?

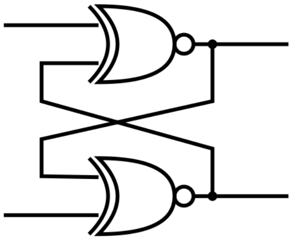


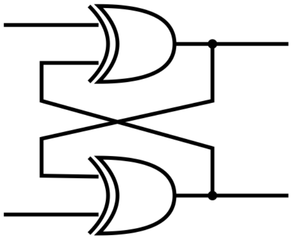
   - ○ A. D flip flop, edge triggered
   - ○ B. D-R latch
   - ○ C. D latch, level triggered
   - ○ D. S-R latch, active high
   - ○ E. S-R latch, active low

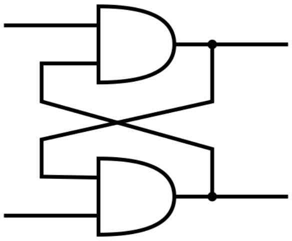
7. Which of the following circuits can form a latch?

○ A. 

○ B. 

○ C. 

○ D. 

○ E. 

Integers

8. What is 1230_4 in base 32? Assume $A_{32} = 10$, $B_{32} = 11$, ... , $G_{32} = 16$, etc.
   - ○ A. $3C_{32}$
   - ○ B. $3D_{32}$
   - ○ C. BT_{32}
   - ○ D. $3C0_{32}$
   - ○ E. $4D_{32}$

9. What is the range of values for an integer in 8-bit sign-magnitude representation?
   - ○ A. -127 to 128
   - ○ B. -127 to 127
   - ○ C. 0 to 255
   - ○ D. -128 to 127
   - ○ E. -128 to 128

10. Extend the following 4-bit sign-magnitude value to 8-bits: `0b1101`
    - ○ A. `0b11111101`
    - ○ B. `0b00001101`
    - ○ C. `0b10001101`
    - ○ D. `0b10000101`
    - ○ E. `0b00001101`

11. What is the decimal equivalent of the 8-bit two's complement number `0b10010111`?
    - ○ A. -105
    - ○ B. -151
    - ○ C. 151
    - ○ D. 105
    - ○ E. -104

12. Convert 210_3 to base 5.
    - ○ A. 21_5
    - ○ B. 41_{10}
    - ○ C. 210_5
    - ○ D. 211_5
    - ○ E. 41_5

13. What is the lowest number that can be represented using 8-bit bias 127 representation?
    - ○ A. 127
    - ○ B. -127
    - ○ C. -256
    - ○ D. 0
    - ○ E. -128

14. Convert the 8-bit two's complement number `0b11001101` to 8-bit sign-magnitude representation.
    - ○ A. `0b11001100`
    - ○ B. `0b01001100`
    - ○ C. `0b00110011`
    - ○ D. `0b01001101`
    - ○ E. `0b10110011`

15. What is the largest unsigned integer a 6-bit register can hold?
  - ○ A. 0x8
  - ○ B. 0xF
  - ○ C. 0xFF
  - ○ D. 0xFFF
  - ○ E. 0x3F

Fractions & Floating Point

16. Which IEEE 754 single precision floating point number is furthest from zero?
  - ○ A. 0x4479C000
  - ○ B. 0xC47A0000
  - ○ C. 0x41300000
  - ○ D. 0xC25C0000
  - ○ E. 0x431B0000

17. Convert the decimal value 51.8_{10} to unsigned fractional binary
  - ○ A. $110011.\overline{1100}$
  - ○ B. 110011.0001
  - ○ C. 110011.1000
  - ○ D. 110011.1100
  - ○ E. $110011.\overline{0001}$

18. Which IEEE 754 single precision floating point number has the largest positive exponent?
  - ○ A. 0x42903333
  - ○ B. 0x43F7999A
  - ○ C. 0xC3018000
  - ○ D. 0xC2366666
  - ○ E. 0x425A6666

19. Convert the floating point number 0x40400000 to unsigned binary.
  - ○ A. 0b101
  - ○ B. 0b001
  - ○ C. 0b011
  - ○ D. 0b110
  - ○ E. 0b010

Strings

20. What is printed to the screen in this MIPS program?

```
.data
P1: .space 27
P2: .asciiz "ABCDEFGHIJKLMNOPQRSTUZWXYZ"

.text
L1:   la   $t0, P1
      addi $t1, $zero, 26
      addi $t2, $zero, 97    # ascii value for ’a’

L2:   sb   $t2, ($t0)
      addi $t1, $t1,   -1
      beqz $t1, GLUE
      addi $t0, $t0,    1    # increment address
      addi $t2, $t2,    1    # increment ascii value
      b    L2

GLUE: li   $v0, 4
      la   $a0, P1
      syscall

      li   $v0, 10
      syscall
```

○ A. abcdefghijklmnopqrstuvwxyz
○ B. ABCDEFGHIJKLMNOPQRSTUZWXYZ
○ C. Correct answer not listed; runtime error
○ D. abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUZWXYZ
○ E. 27

21. Decode the following ASCII string. Values are given in hex:
`49 20 68 61 76 65 20 74 68 65 20 68 69 67 68 20 67 72 6f 75 6e 64 21.`

○ A. I have the high ground!
○ B. I have no idea what the other sentences mean.
○ C. It’s over Anakin!
○ D. You underestimate my power!
○ E. Don’t try it.

Arithmetic & Logical Operations

22. What is the result of a bit-wise XOR performed on the following 8-bit binary numbers:

```
  0b 1 0 1 1 0 1 1 0
⊕ 0b 1 0 1 0 1 0 1 0
_________________________
```

- ○ A. 0b01000001
- ○ B. 0b00011100
- ○ C. 0b10111110
- ○ D. 0b11100011
- ○ E. 0b10100010

23. What is the result of a shift right arithmetic by three and a shift right logical by three of the 8-bit number 10010110 = 0x96? The operations are performed independently of each other.

- ○ A. 0x12 and 0x12
- ○ B. 0xB0 and 0xB7
- ○ C. 0x12 and 0xF2
- ○ D. 0xB7 and 0xB0
- ○ E. 0xF2 and 0x12

24. Which of these 8-bit two's complement computations has carry out but no overflow? Select all that apply.

- ○ A. 0x80 + 0x80 = 0x00
- ○ B. 0xFB + 0xCC = 0xC7
- ○ C. 0x7F + 0x70 = 0xEF
- ○ D. 0x89 + 0xFF = 0x88
- ○ E. 0xA7 + 0x61 = 0x08

Memory

25. Assume a little endian memory system. What is stored in $s0 after the following program is executed?

```
.data
flux:           .word  0xC0FFEEEE
some_data:      .byte  0xFE 0xED 0xBB
some_more_data: .byte  0xCE    1    2 0x00

.text
la  $t1   some_more_data
lw  $t0   ($t1)
sb  $t0  2($t1)
lw  $s0   ($t1)
```

- ○ A. 0x00CE01CE
- ○ B. 0x000200CE
- ○ C. Answer not listed; memory alignment error
- ○ D. 0xCE010000
- ○ E. 0xCE01CE00

26. How many bits are needed to represent the address in a byte-addressable memory space with capacity of 5TB?
    - ○ A. 43
    - ○ B. Correct answer not listed
    - ○ C. 33
    - ○ D. 20
    - ○ E. 40

27. How many 32-bit integers can be stored in the array labeled `myArray` as shown below:

```
.data
msg:       .asciiz "Good luck!!"
myArray:   .space 20
tacos:     .asciiz "Tacos and 2SC make me happy!!"
```

- ○ A. 80
- ○ B. 5
- ○ C. 4
- ○ D. 10
- ○ E. 2.5

MIPS Instruction Set Architecture

28. How can we create a mask for bits 4:14 of `$t0`?
    - ○ A. `andi $t0 $t0 0x7ff0`
    - ○ B. `andi $t0 $t0 0x800f`
    - ○ C. `ori  $t0 $t0 0x800f`
    - ○ D. `ori  $t0 $t0 0x7ff0`
    - ○ E. `xori $t0 $t0 0x7ff0`

29. What is the value in `$10` after the following instructions are executed?

```
ADDI  $10 $0  11
SLL   $10 $10 30
SRL   $10 $10 29
```

- ○ A. `0xFFFE`
- ○ B. `0xFFFF`
- ○ C. `0x000B`
- ○ D. `0x000F`
- ○ E. `0x000E`

30. Decode the following MIPS instruction. Select all that apply.

```
0x8D090008
```

- ○ A. `sw    $8   8($9)`
- ○ B. `addi  $8     $9    8`
- ○ C. `lw    $t1  8($t0)`
- ○ D. `sw    $t1  8($t0)`
- ○ E. `lw    $t0  8($t1)`

31. Assume `$s0=0x6` and `$t7=0xA`. What value is stored in `$t7` after the following instruction?

```
div $t7 $s0
```

○ A. `0x1`
○ B. `0x6`
○ C. `0x4`
○ D. `0x0`
○ E. `0xA`

32. Decode the following MIPS instruction. Select all that apply.

```
0x012F4020
```

○ A. `ADD $8  $9   $15`
○ B. `AND $9  $15  $8`
○ C. `ADD $t1 $t7  $t0`
○ D. `ADD $t0 $t1  $t7`
○ E. `ADD $9  $15  $8`

33. What is the size of a register in MIPS32? Select all that apply.

○ A. 64 bits
○ B. 8 bytes
○ C. 32 bits
○ D. 8 nybbles
○ E. 4 bytes

34. What is the value in `$t0` after the following instructions are executed?

```
li  $t0, 5
li  $t1, 10
xor $t0, $t0, $t0

loop: nop
addi  $t0, $t0,  1
subi  $t1, $t1,  1
bgtz  $t1, loop

li $v0, 10
syscall
```

○ A. 16
○ B. 15
○ C. 10
○ D. 5
○ E. 0

35. What is the value of register `$v0` after the following instructions?

```
    addi $t1 $zero 8
    addi $s0 $zero 50       # 50 = 0b110010
    addi $v0 $zero 0
loop: nop
    andi $a0  $s0  0
    add  $v0  $v0  $a0
    srl  $t1  $t1  1
    bnez $t1  loop
```

○ A. 2
○ B. 20
○ C. 18
○ D. 0
○ E. 50

Stack & Subroutines

36. Which instruction will the program counter point to after the “jr $ra” instruction executes in the Prompt_user subroutine?

```
.data
P1: .asciiz "Input: "
N1: .word

.text
      la  $a0, P1
      la  $a1, N1
      jal Prompt_user

halt: li  $v0, 10
      syscall

PrintString:
      li   $v0, 4
      syscall
      jr   $ra

Prompt_user:
      jal  PrintString
      move $a0, $a1
      li   $v0, 8
      syscall
      jr   $ra
```

○ A. `jal Prompt_user`
○ B. `jal PrintString`
○ C. `move $a0, $a1`
○ D. Answer not listed; code doesn’t assemble
○ E. `halt:  li $v0, 10`

37. Which combination of MIPS instructions perform a push operation of two elements (in $t0 and $t1) on the stack? Select all that apply.

○ A.
```
sw   $t0,  ($sp)
sw   $t1, 4($sp)
subi $sp,   $sp, 8
```

○ B.
```
subi $sp,   $sp, 8
sw   $t0,  ($sp)
sw   $t1, 4($sp)
```

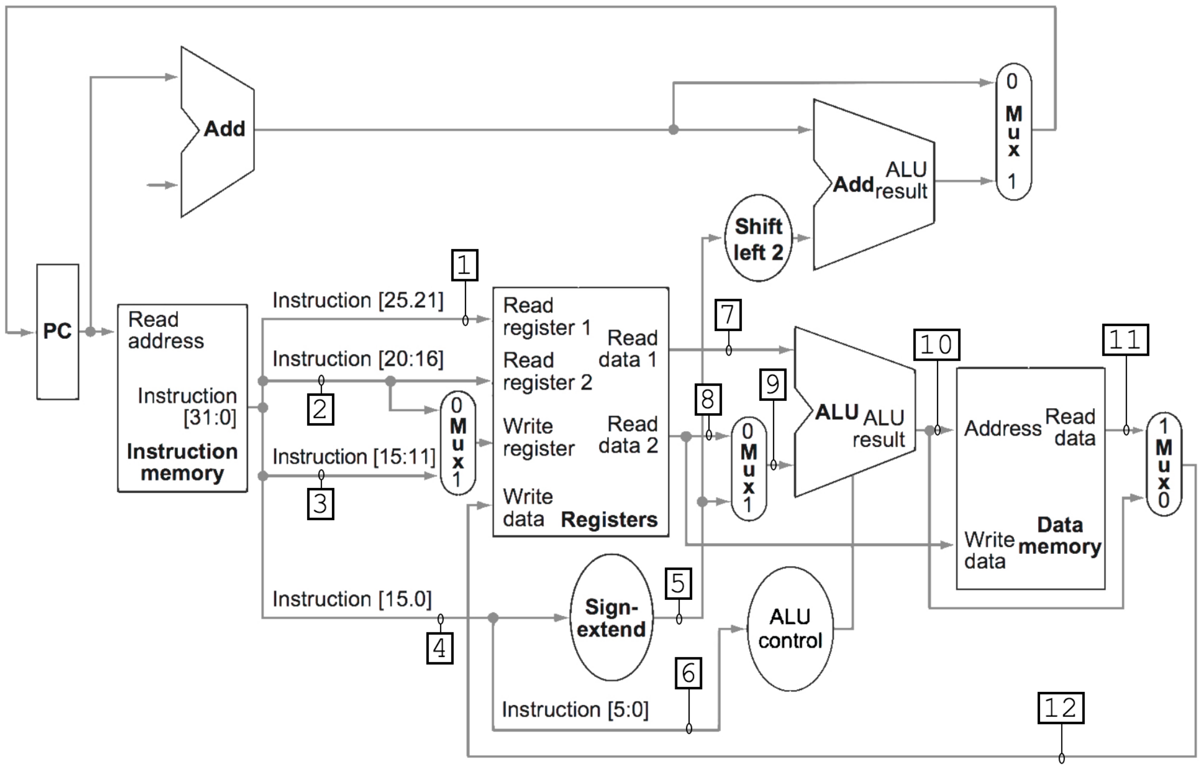
○ C.
```
subi $sp,   $sp, 4
sw   $t0,  ($sp)
subi $sp,   $sp, 4
sw   $t1,  ($sp)
```

○ D.
```
lw   $t0,  ($sp)
lw   $t1,  ($sp)
addi $sp,   $sp, 8
```

○ E.
```
addi $sp,   $sp, 4
lw   $t0,  ($sp)
addi $sp,   $sp, 4
lw   $t1,  ($sp)
```

Data Path

Refer to this MIPS data path for the next three questions:



38. Assume $s0 = 0xAB, $s1 = 0x11 and SH $s1 8($s0) is executed. What is the value on wire '8'?
   - ○ A. Not enough information given.
   - ○ B. 0x11
   - ○ C. 0xAB
   - ○ D. 0x08
   - ○ E. 0x10

39. Assume instruction 0x150802C3 is executed. What is the value on wire '4'?
   - ○ A. 0x0B0C
   - ○ B. 0x10
   - ○ C. Not enough information given.
   - ○ D. 0x02C3
   - ○ E. 0x11

40. Assume the values on wires '1', '5', '10', '11' and '12' are 0x08, 0x10, 0xAF, 0xBE and 0xBE respectively. Which instruction could correspond to these values?
   - ○ A. LW $s0 16($s0)
   - ○ B. ADDI $t0 $t0 0x10
   - ○ C. LB $t1 16($t0)
   - ○ D. LH $7 10($8)
   - ○ E. Not enough information given.

Command Line Interface

41. True or False: Listing the files of a different directory changes the directory you are in.
    - ○ A. False
    - ○ B. True

42. True or False: The command 'mv' can be used to rename a file.
    - ○ A. True
    - ○ B. False

REG NAME	REG #	MNEMONIC	MEANING	TYPE	OPCODE	FUNCT	MNEMONIC	MEANING	TYPE	OPCODE	FUNCT
$zero	0	sll	Logical Shift Left	R	0x00	0x00	add	Add	R	0x00	0x20
$at	1	srl	Logical Shift Right (0-extended)	R	0x00	0x02	addi	Add Immediate	I	0x08	NA
$v0	2	sra	Arithmetic Shift Right (sign-extended)	R	0x00	0x03	addiu	Add Unsigned Immediate	I	0x09	NA
$v1	3	jr	Jump to Address in Register	R	0x00	0x08	addu	Add Unsigned	R	0x00	0x21
$a0	4	mfhi	Move from HI Register	R	0x00	0x10	and	Bitwise AND	R	0x00	0x24
$a1	5	mflo	Move from LO Register	R	0x00	0x12	andi	Bitwise AND Immediate	I	0x0C	NA
$a2	6	mult	Multiply	R	0x00	0x18	beq	Branch if Equal	I	0x04	NA
$a3	7	multu	Unsigned Multiply	R	0x00	0x19	blez	Branch if Less Than or Equal to Zero	I	0x06	NA
$t0	8	div	Divide	R	0x00	0x1A	bne	Branch if Not Equal	I	0x05	NA
$t1	9	divu	Unsigned Divide	R	0x00	0x1B	div	Divide	R	0x00	0x1A
$t2	10	add	Add	R	0x00	0x20	divu	Unsigned Divide	R	0x00	0x1B
$t3	11	addu	Add Unsigned	R	0x00	0x21	j	Jump to Address	J	0x02	NA
$t4	12	sub	Subtract	R	0x00	0x22	jal	Jump and Link	J	0x03	NA
$t5	13	subu	Unsigned Subtract	R	0x00	0x23	jr	Jump to Address in Register	R	0x00	0x08
$t6	14	and	Bitwise AND	R	0x00	0x24	lb	Load Byte	I	0x20	NA
$t7	15	or	Bitwise OR	R	0x00	0x25	lbu	Load Byte Unsigned	I	0x24	NA
$s0	16	xor	Bitwise XOR (Exclusive-OR)	R	0x00	0x26	lh	Load Halfword	I	0x21	NA
$s1	17	nor	Bitwise NOR (NOT-OR)	R	0x00	0x27	lhu	Load Halfword Unsigned	I	0x25	NA
$s2	18	slt	Set to 1 if Less Than	R	0x00	0x2A	lui	Load Upper Immediate	I	0x0F	NA
$s3	19	sltu	Set to 1 if Less Than Unsigned	R	0x00	0x2B	lw	Load Word	I	0x23	NA
$s4	20	j	Jump to Address	J	0x02	NA	mfc0	Move from Coprocessor 0	R	0x10	NA
$s5	21	jal	Jump and Link	J	0x03	NA	mfhi	Move from HI Register	R	0x00	0x10
$s6	22	beq	Branch if Equal	I	0x04	NA	mflo	Move from LO Register	R	0x00	0x12
$s7	23	bne	Branch if Not Equal	I	0x05	NA	mult	Multiply	R	0x00	0x18
$t8	24	blez	Branch if Less Than or Equal to Zero	I	0x06	NA	multu	Unsigned Multiply	R	0x00	0x19
$t9	25	addi	Add Immediate	I	0x08	NA	nor	Bitwise NOR (NOT-OR)	R	0x00	0x27
$k0	26	addiu	Add Unsigned Immediate	I	0x09	NA	or	Bitwise OR	R	0x00	0x25
$k1	27	slti	Set to 1 if Less Than Immediate	I	0x0A	NA	ori	Bitwise OR Immediate	I	0x0D	NA
$gp	28	sltiu	Set to 1 if Less Than Unsigned Immediate	I	0x0B	NA	sb	Store Byte	I	0x28	NA
$sp	29	andi	Bitwise AND Immediate	I	0x0C	NA	sh	Store Halfword	I	0x29	NA
		ori	Bitwise OR Immediate	I	0x0D	NA	sll	Logical Shift Left	R	0x00	0x00
		xori	Bitwise XOR (Exclusive-OR) Immediate	I	0x0E	NA	slt	Set to 1 if Less Than	R	0x00	0x2A
		lui	Load Upper Immediate	I	0x0F	NA	slti	Set to 1 if Less Than Immediate	I	0x0A	NA
		mfc0	Move from Coprocessor 0	R	0x10	NA	sltiu	Set to 1 if Less Than Unsigned Immediate	I	0x0B	NA
		lb	Load Byte	I	0x20	NA	sltu	Set to 1 if Less Than Unsigned	R	0x00	0x2B
		lh	Load Halfword	I	0x21	NA	sra	Arithmetic Shift Right (sign-extended)	R	0x00	0x03
		lw	Load Word	I	0x23	NA	srl	Logical Shift Right (0-extended)	R	0x00	0x02
		lbu	Load Byte Unsigned	I	0x24	NA	sub	Subtract	R	0x00	0x22
		lhu	Load Halfword Unsigned	I	0x25	NA	subu	Unsigned Subtract	R	0x00	0x23
		sb	Store Byte	I	0x28	NA	sw	Store Word	I	0x2B	NA
		sh	Store Halfword	I	0x29	NA	xor	Bitwise XOR (Exclusive-OR)	R	0x00	0x26
		sw	Store Word	I	0x2B	NA	xori	Bitwise XOR (Exclusive-OR) Immediate	I	0x0E	NA

```
R Type:  instr rd rs rt    (arithmetic, logical)
         instr rd rt shamt (shifts)
```

31 26	25 21	20 16	15 11	10 6	5 0
<- 6 bits ->	<- 5 bits ->	<- 5 bits ->	<- 5 bits ->	<- 5 bits ->	<- 6 bits ->
opcode	rs	rt	rd	shamt	funct

```
I Type:  instr  rt rs immediate    (arithmetic, logical)
         branch rs rt immediate    (branches)
         instr  rt immediate(rs)   (loads, stores)
```

31 26	25 21	20 16	15 0
<- 6 bits ->	<- 5 bits ->	<- 5 bits ->	<- 16 bits ->
opcode	rs	rt	immediate

```
J Type:  j immediate (jumps)
```

31 26	25 0
<- 6 bits ->	<- 26 bits ->
opcode	immediate

ASCII CODE						ASCII CODE						ASCII CODE					-	ASCII CODE				
BIN	OCT	DEC	HEX	CHARACTER		BIN	OCT	DEC	HEX	CHARACTER		BIN	OCT	DEC	HEX	CHARACTER		BIN	OCT	DEC	HEX	CHARACTER
010 0000	40	32	20	space		011 1000	70	56	38	8		101 0000	120	80	50	P		110 1000	150	104	68	h
010 0001	41	33	21	!		011 1001	71	57	39	9		101 0001	121	81	51	Q		110 1001	151	105	69	i
010 0010	42	34	22	"		011 1010	72	58	3A	:		101 0010	122	82	52	R		110 1010	152	106	6A	j
010 0011	43	35	23	#		011 1011	73	59	3B	;		101 0011	123	83	53	S		110 1011	153	107	6B	k
010 0100	44	36	24	$		011 1100	74	60	3C	<		101 0100	124	84	54	T		110 1100	154	108	6C	l
010 0101	45	37	25	%		011 1101	75	61	3D	=		101 0101	125	85	55	U		110 1101	155	109	6D	m
010 0110	46	38	26	&		011 1110	76	62	3E	>		101 0110	126	86	56	V		110 1110	156	110	6E	n
010 0111	47	39	27	'		011 1111	77	63	3F	?		101 0111	127	87	57	W		110 1111	157	111	6F	o
010 1000	50	40	28	(		100 0000	100	64	40	@		101 1000	130	88	58	X		111 0000	160	112	70	p
010 1001	51	41	29	)		100 0001	101	65	41	A		101 1001	131	89	59	Y		111 0001	161	113	71	q
010 1010	52	42	2A	*		100 0010	102	66	42	B		101 1010	132	90	5A	Z		111 0010	162	114	72	r
010 1011	53	43	2B	+		100 0011	103	67	43	C		101 1011	133	91	5B	[		111 0011	163	115	73	s
010 1100	54	44	2C	,		100 0100	104	68	44	D		101 1100	134	92	5C	\		111 0100	164	116	74	t
010 1101	55	45	2D	-		100 0101	105	69	45	E		101 1101	135	93	5D	]		111 0101	165	117	75	u
010 1110	56	46	2E	.		100 0110	106	70	46	F		101 1110	136	94	5E	^		111 0110	166	118	76	v
010 1111	57	47	2F	/		100 0111	107	71	47	G		101 1111	137	95	5F	_		111 0111	167	119	77	w
011 0000	60	48	30	0		100 1000	110	72	48	H		110 0000	140	96	60	`		111 1000	170	120	78	x
011 0001	61	49	31	1		100 1001	111	73	49	I		110 0001	141	97	61	a		111 1001	171	121	79	y
011 0010	62	50	32	2		100 1010	112	74	4A	J		110 0010	142	98	62	b		111 1010	172	122	7A	z
011 0011	63	51	33	3		100 1011	113	75	4B	K		110 0011	143	99	63	c		111 1011	173	123	7B	{
011 0100	64	52	34	4		100 1100	114	76	4C	L		110 0100	144	100	64	d		111 1100	174	124	7C	|
011 0101	65	53	35	5		100 1101	115	77	4D	M		110 0101	145	101	65	e		111 1101	175	125	7D	}
011 0110	66	54	36	6		100 1110	116	78	4E	N		110 0110	146	102	66	f		111 1110	178	126	7E	~
011 0111	67	55	37	7		100 1111	117	79	4F	O		110 0111	147	103	67	g		111 1111	177	127	7F	DEL

Note: ASCII codes 0x00 -> 0x1F are unprintable control characters used to control peripherals (e.g. printers)

SERVICE	CODE IN $v0	ARGUMENTS	RESULT
print integer	1	$a0 = integer to print	
print float	2	$f12 = float to print	
print double	3	$f12 = double to print	
print string	4	$a0 = address of null-terminated string to print	
read integer	5		$v0 contains integer read
read float	6		$f0 contains float read
read double	7		$f0 contains double read
read string	8	$a0 = address of input buffer $a1 = maximum number of characters to read	*See note below table*
sbrk (allocate heap memory)	9	$a0 = number of bytes to allocate	$v0 contains address of allocated memory
exit (terminate execution)	10		
print character	11	$a0 = character to print	*See note below table*
read character	12		$v0 contains character read
open file	13	$a0 = address of null-terminated string containing filename $a1 = flags $a2 = mode	$v0 contains file descriptor (negative if error). See note below table
read from file	14	$a0 = file descriptor $a1 = address of input buffer $a2 = maximum number of characters to read	$v0 contains number of characters read (0 if end-of-file, negative if error). See note below table
write to file	15	$a0 = file descriptor $a1 = address of output buffer $a2 = number of characters to write	$v0 contains number of characters written (negative if error). See note below table
close file	16	$a0 = file descriptor	
exit2 (terminate with value)	17	$a0 = termination result	*See note below table*
Services 1 through 17 are compatible with the SPIM simulator, other than Open File (13) as described in the Notes below the table. Services 30 and higher are exclusive to MARS.			
time (system time)	30		$a0 = low order 32 bits of system time $a1 = high order 32 bits of system time. See note below table
MIDI out	31	$a0 = pitch (0-127) $a1 = duration in milliseconds $a2 = instrument (0-127) $a3 = volume (0-127)	Generate tone and return immediately. See note below table
sleep	32	$a0 = the length of time to sleep in milliseconds.	Causes the MARS Java thread to sleep for (at least) the specified number of milliseconds. This timing will not be precise, as the Java implementation will add some overhead.
MIDI out synchronous	33	$a0 = pitch (0-127) $a1 = duration in milliseconds $a2 = instrument (0-127) $a3 = volume (0-127)	Generate tone and return upon tone completion. See note below table
print integer in hexadecimal	34	$a0 = integer to print	Displayed value is 8 hexadecimal digits, left-padding with zeroes if necessary.
print integer in binary	35	$a0 = integer to print	Displayed value is 32 bits, left-padding with zeroes if necessary.
print integer as unsigned	36	$a0 = integer to print	Displayed as unsigned decimal value.