Final Exam - Fall 2019

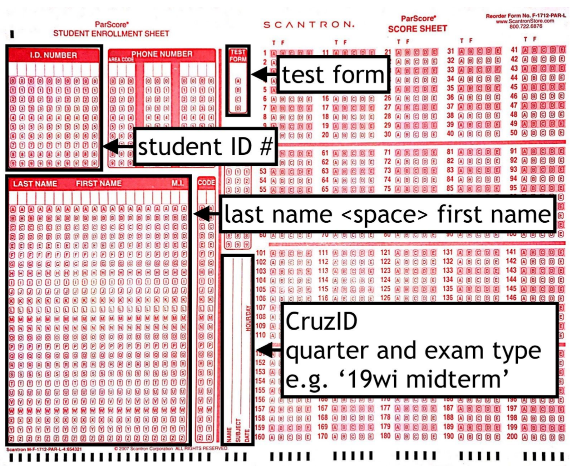
CSE 12: Computer Systems and Assembly Language

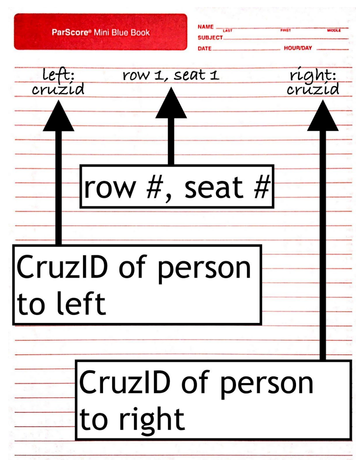
University of California, Santa Cruz

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 on page 1). In the center of the Scantron form 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. If you are taking the exam in a DRC room, write "DRC" for the row and seat number. Once you are permitted to begin, write your CruzID on all subsequent pages of the exam. If there is no one to your left or right, write "empty" or "aisle."

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 completely OFF.

There are 35 questions on this exam; you only need to answer 33 for full points. The additional 2 questions (of your choosing) will be counted as extra credit. All questions are multiple choice, and some questions might 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 ____________

CSE 12 Final - Version A

Fall 2019

Integer Numbering Systems

1. Assume a base 32 integer numbering system that includes all hexadecimal digits, and G=16, H=17, ... R=27, ... V=31. Convert the base 32 number R2D2 to base 16.
   - ◯ A. `0x1B2D2`
   - ◯ B. I forgot the shortcut to do this problem
   - ◯ C. `0x3304642`
   - ◯ D. `0xD89A2`
   - ◯ E. `0x1B020D02`
2. Convert the base 6 number 102_6 to base 3.
   - ◯ A. 10002_3
   - ◯ B. 1102_3
   - ◯ C. 15_3
   - ◯ D. 38_3

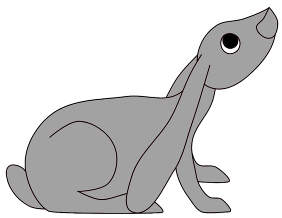
Boolean Algebra

3. Select three equivalent Boolean expressions.
   - ◯ A. A
   - ◯ B. $\bar{A}B$
   - ◯ C. $\bar{A}$
   - ◯ D. $\overline{AB}(\bar{A}+B)$
   - ◯ E. $(\bar{A}+\bar{B})(\bar{A}+B)$

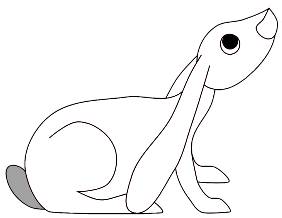
4. Assume each rabbit listed below corresponds to a minterm where B means a gray body and T means a gray tail. For instance, the all white rabbit corresponds with minterm $\bar{B}\bar{T}$.

The following Boolean expression includes which rabbits (i.e. which minterms would be listed as 1 on a truth table)?

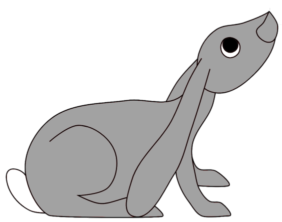
$$B + T + \overline{BT}$$



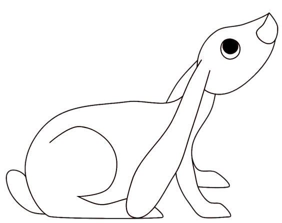
○ A.



○ B.



○ C.

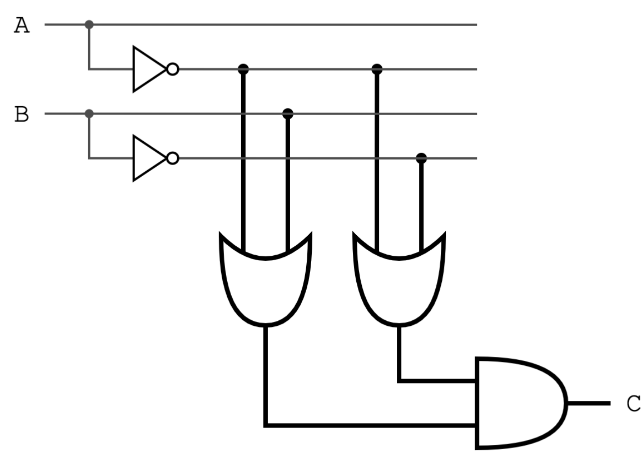


○ D.

5. Select all equivalent expressions.
   - ○ A. $A + C$
   - ○ B. $AA + AC + C$
   - ○ C. $A(A + C) + C$
   - ○ D. $(A + C)A + AC + C$
   - ○ E. $(A + C)(AD + A\bar{D}) + AC + C$

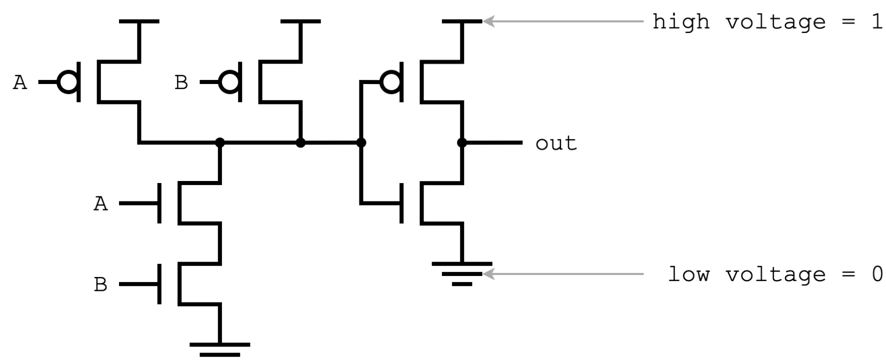
Combinational Logic

6. Which equation describes this circuit?



- ◯ A. Answer not listed
- ◯ B. $C = A$
- ◯ C. $C = \bar{A}$
- ◯ D. $C = (A + B)(A + \bar{B})$
- ◯ E. $C = \bar{A}\bar{B} + \bar{A}B$

7. This circuit represents which logic element?



- ◯ A. NAND gate
- ◯ B. XNOR gate
- ◯ C. NOR gate
- ◯ D. OR gate
- ◯ E. AND gate

8. What is the **product of sums** solution to this truth table?

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

◯ A. D = $\bar{A}\bar{B}C+\bar{A}B\bar{C}+\bar{A}BC+ABC$

◯ B. D = $\bar{A}\bar{B}\bar{C}+A\bar{B}\bar{C}+A\bar{B}C+AB\bar{C}$

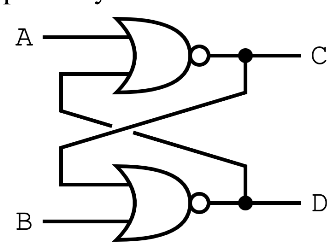
◯ C. D = $AB\bar{C}+A\bar{B}C+A\bar{B}\bar{C}+\bar{A}\bar{B}\bar{C}$

◯ D. D = $(A+B+C)(\bar{A}+B+C)(\bar{A}+B+\bar{C})(\bar{A}+\bar{B}+C)$

◯ E. D = $(\bar{A}+\bar{B}+\bar{C})(A+\bar{B}+\bar{C})(A+\bar{B}+C)(A+B+\bar{C})$

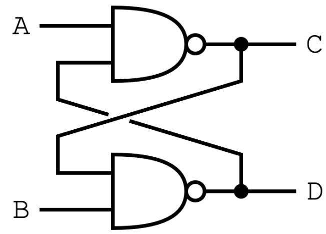
Sequential Logic

9. Assume values A and B are 1 and 1, respectively. What are the values on wires C and D, respectively?



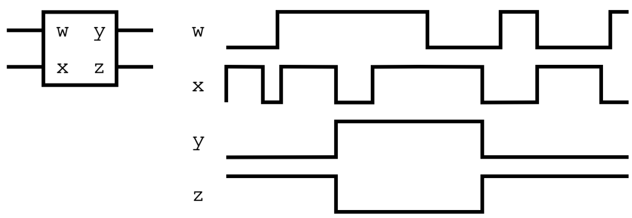
◯ A. `0, 1`
◯ B. `1, 0`
◯ C. `1, 1`
◯ D. There is not enough information to answer
◯ E. `0, 0`

10. Assume values A and B are 1 and 0, respectively. What are the values on wires C and D, respectively?



- ◯ A. `1, 0`
- ◯ B. `0, 0`
- ◯ C. `1, 1`
- ◯ D. There is not enough information to answer
- ◯ E. `0, 1`

11. What device does this timing diagram represent?



- ◯ A. D Flip-Flop, falling edge triggered
- ◯ B. D Latch, level triggered
- ◯ C. D Flip-Flop, rising edge triggered
- ◯ D. S-R Latch, active low
- ◯ E. D-R Latch

Fractions

12. Convert the IEEE 754 single precision floating point number 0xC0200000 to decimal.
- ◯ A. `0.5`
- ◯ B. Answer not listed
- ◯ C. `-2.5`
- ◯ D. `-1.5`
- ◯ E. `1.5`

13. Convert the 3 + 5 fixed point binary number 11001010 to decimal.
- ◯ A. `25.25`
- ◯ B. `12.625`
- ◯ C. `6.1`
- ◯ D. Answer not listed
- ◯ E. `6.3125`

14. Express 13.4 in 6 + 10 fixed point binary notation.
   - ◯ A. `0x0359`
   - ◯ B. `0xD666`
   - ◯ C. `0xD400`
   - ◯ D. `0x3599`
   - ◯ E. `0x3500`

15. Extend the 4-bit two's complement number `0xA` to 8-bit two's complement.
   - ◯ A. Answer not listed
   - ◯ B. `0xFA`
   - ◯ C. `0x0A`
   - ◯ D. `0x12`
   - ◯ E. `0x1A`

Arithmetic & Logical Operations

16. What is the result of bitwise XOR between `0xAB` and `0xEE`?
   - ◯ A. `0xBA`
   - ◯ B. `0`
   - ◯ C. `0xEF`
   - ◯ D. `0x45`
   - ◯ E. `0xAA`

17. What is the result after logical shift left `0xD011` by 4?
   - ◯ A. Answer not listed
   - ◯ B. `0x11D0`
   - ◯ C. `0x0`
   - ◯ D. `0x11FF`
   - ◯ E. `0x11`

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

◯ A.
```
  1 0 1 0 1 0 1 1
+ 0 1 0 1 0 1 0 1
```

◯ B.
```
  0 1 1 1 1 1 1 1
+ 0 0 0 0 0 0 0 1
```

◯ C.
```
  1 0 0 0 0 0 0 0
+ 1 1 1 1 1 1 1 1
```

◯ D.
```
  1 1 1 1 0 1 1 0
+ 1 1 0 1 1 0 0 0
```

◯ E.
```
  1 0 1 0 0 1 0 0
+ 0 1 1 0 1 1 1 0
```

Addressability

19. Assume a 4TB memory space with 2^{18} memory locations. How many bytes of data are stored at each memory location?
   - ◯ A. Answer not listed
   - ◯ B. 16KB
   - ◯ C. 24GB
   - ◯ D. 16MB
   - ◯ E. 2.22GB

Data & Syscalls

20. After running the following instructions, assume `"ABCD"` is printed to screen.

```
la $a0 str
li $v0 4
syscall
```

What is printed if running the following instructions?

```
li $t0 0x45464748
sh $t0 ($a0)
syscall
```

- ◯ A. `HGCD`
- ◯ B. There is not enough information to answer.
- ◯ C. `ABCE`
- ◯ D. `ABCD`
- ◯ E. `ABEF`

21. After running the following instructions, assume `"0x41424344"` is printed to screen.

```
move $a0 $t0
li   $v0  34
syscall
```

What is printed if running the following instructions?

```
li $v0 11
syscall
```

- ◯ A. Answer not listed
- ◯ B. `A`
- ◯ C. `0x44`
- ◯ D. `D`
- ◯ E. `0x41`

22. What is printed to the screen after the following program?

```
.data
string: .byte 0x49 0x20 0x61 0x6D 0x20 0x79 0x6F 0x75 0x72
        .byte 0x20 0x66 0x61 0x74 0x68 0x65 0x72 0x00

.text
la $a0 string
li $v0 4
syscall
```

- ◯ A. `You were my brother`
- ◯ B. `I am your flux`
- ◯ C. `I am your father`
- ◯ D. `I am your friend`
- ◯ E. `You're my only hope`

23. Which data allocation takes the most memory space? Select all that apply.

- ◯ A. `.ascii  "HelloWorld\n"`
- ◯ B. `.asciiz "HelloWorld"`
- ◯ C. `.word   0xABCD 0x12345678`
- ◯ D. `.float  1500.5 1000.11111`
- ◯ E. `.space  10`

For the following three problems, assume the static data of the program is given as such:

```
.data
flux:  .byte  0x46 0x6c 0x75 0x78 0x20
bunny: .ascii "Bunny"
heart: .word  0x333c20
```

24. What is printed to the screen after the following instructions?

```
.text
addi $v0 $zero 4
la   $a0 flux
syscall
```

- ◯ A. `Flux Bunny <3`
- ◯ B. `Flux`
- ◯ C. `Flux Bunny`
- ◯ D. `Flux Bunny<3`
- ◯ E. `Flux Bunny3<`

25. Assume the label ‘flux’ represents address 0x10010000. What is stored in $t6 after the following instruction?

```
.text
la $t6 heart
```

- ◯ A. `0x10010011`
- ◯ B. `0x1001000A`
- ◯ C. `0x1001000B`
- ◯ D. `0x10010010`
- ◯ E. `0x1001000C`

26. What is printed to the screen after the following instructions?

```
.text
addiu $v0 $zero 34
la    $s4 bunny
lw    $a0 ($s4)
syscall
```

- ◯ A. `0x796e6e75`
- ◯ B. `0x42756e6e`
- ◯ C. Nothing; memory alignment error
- ◯ D. `0x756e6e79`
- ◯ E. `0x6e6e7542`

For the next three questions, consider the following little endian memory system. Note the order of the addresses listed.

Address	Data
0x00	0xA0
0x01	0xB0
0x02	0xC0
0x03	0xD0
0x04	0x12
0x05	0x34
0x06	0x56
0x07	0x78

27. What is the value in $s1 after running the following instructions?

```
addi $s0  $zero 0x04
lw   $s1  ($s0)
```

- ◯ A. 0x12345678
- ◯ B. 0x12D0C0B0
- ◯ C. Answer not listed
- ◯ D. 0x87654321
- ◯ E. 0x78563412

28. What is the value in $s1 after running the following instructions?

```
sub $s0    $s0   $s0
lb  $t0  1($s0)
sh  $t0  2($s0)
lw  $s1   ($s0)
```

- ◯ A. Answer not listed
- ◯ B. 0xA0B0FFB0
- ◯ C. 0xFFB0B0A0
- ◯ D. 0xA0B000B0
- ◯ E. Unknown; memory alignment error

29. What is the value in $s1 after running the following instructions?

```
addi $s0  $zero    0x04
lh   $s1  -2($s0)
```

- ◯ A. 0xFFFFC0D0
- ◯ B. 0x0000D0C0
- ◯ C. Answer not listed
- ◯ D. 0xFFFFD0C0
- ◯ E. 0x0000C0D0

Stack & Subroutines

30. Given the following instructions and corresponding memory addresses, what is the value in $ra after running the instruction `jal label`? (label = 0x100C)

Address	Data
`0x1000`	`addi $ra $ra 0x40`
`0x1004`	`jal label`
`0x1008`	`xor $s0 $s0 $s0`
`0x100C`	`label:  jr $ra`

- ◯ A. `0x40`
- ◯ B. `0x1008`
- ◯ C. `0x1004`
- ◯ D. `0x1000`
- ◯ E. `0x100C`

31. Which combination of MIPS instructions performs a pop operation of three elements (`$s0, $s1,` and `$s2`) from the stack?

◯ A.
```
sb    $s0    ($sp)
sb    $s1   1($sp)
sb    $s2   2($sp)
addi  $sp     $sp    3
```

◯ B.
```
lw    $s0    ($sp)
lw    $s1   4($sp)
lw    $s2   8($sp)
addi  $sp     $sp   12
```

◯ C.
```
addi  $sp     $sp   12
sw    $s0    ($sp)
sw    $s1  -4($sp)
sw    $s2  -8($sp)
```

◯ D.
```
addi  $sp     $sp  -12
sw    $s0    ($sp)
sw    $s1   4($sp)
sw    $s2   8($sp)
```

◯ E.
```
lb    $s0    ($sp)
lb    $s1   1($sp)
lb    $s2   2($sp)
addi  $sp     $sp    3
```

Instruction Decoding & Data Path

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

```
0x014B6024
```

- ◯ A. Answer not listed
- ◯ B. `addi $t2 $t3 $t4`
- ◯ C. `and  $10 $11 $12`
- ◯ D. `add  $12 $10 $11`
- ◯ E. `and  $t4 $t2 $t3`

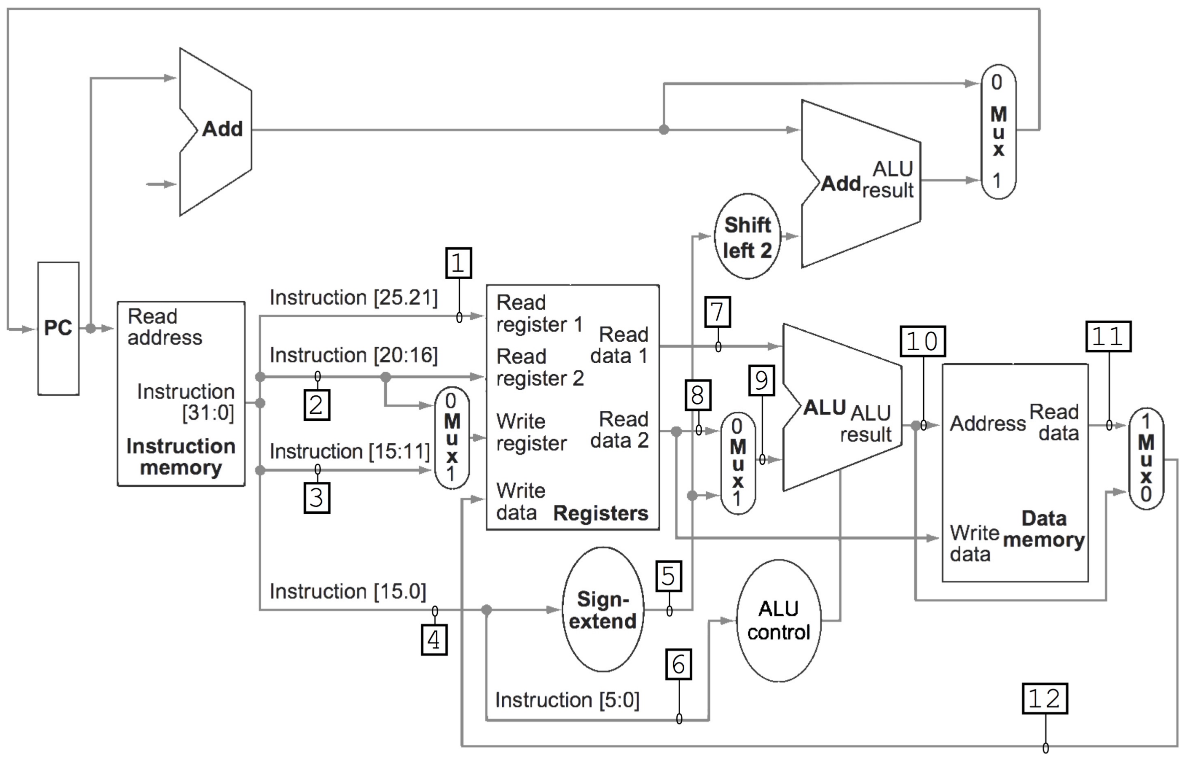
33. Given the branch instruction in machine code

```
000100 00010 01001 1111111111111110
```

assume the branch target address is 0x2004, what is the address of the branch instruction?

- ◯ A. `0x2004`
- ◯ B. Answer not listed
- ◯ C. `0x2014`
- ◯ D. `0x2018`
- ◯ E. `0x2010`

For the next two questions, consider the following data path:



34. Assume \$8 = 0xC, \$9 = 0xD, instruction 0xA10900AB is executed, what is the value on wire '8'?
  - ◯ A. 0xC
  - ◯ B. Not enough information given
  - ◯ C. 0xD
  - ◯ D. 0xAB
  - ◯ E. 0x8

35. Assume the values on wires '2', '10', '11' and '12' are 0x19, 0x20, 0x21 and 0x21 respectively. Which instruction could correspond to these values?
  - ◯ A. `lh $4  32($19)`
  - ◯ B. `sh $19 32($0)`
  - ◯ C. `sb $0  33($s3)`
  - ◯ D. `lw $t9 32($zero)`
  - ◯ E. Not enough information given

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)

REG NAME	REG #
$zero	0
$at	1
$v0	2
$v1	3
$a0	4
$a1	5
$a2	6
$a3	7
$t0	8
$t1	9
$t2	10
$t3	11
$t4	12
$t5	13
$t6	14
$t7	15
$s0	16
$s1	17
$s2	18
$s3	19
$s4	20
$s5	21
$s6	22
$s7	23
$t8	24
$t9	25
$k0	26
$k1	27
$gp	28
$sp	29

MNEMONIC	MEANING	TYPE	OPCODE	FUNCT
sll	Logical Shift Left	R	0x00	0x00
srl	Logical Shift Right (0-extended)	R	0x00	0x02
sra	Arithmetic Shift Right (sign-extended)	R	0x00	0x03
jr	Jump to Address in Register	R	0x00	0x08
mfhi	Move from HI Register	R	0x00	0x10
mflo	Move from LO Register	R	0x00	0x12
mult	Multiply	R	0x00	0x18
multu	Unsigned Multiply	R	0x00	0x19
div	Divide	R	0x00	0x1A
divu	Unsigned Divide	R	0x00	0x1B
add	Add	R	0x00	0x20
addu	Add Unsigned	R	0x00	0x21
sub	Subtract	R	0x00	0x22
subu	Unsigned Subtract	R	0x00	0x23
and	Bitwise AND	R	0x00	0x24
or	Bitwise OR	R	0x00	0x25
xor	Bitwise XOR (Exclusive-OR)	R	0x00	0x26
nor	Bitwise NOR (NOT-OR)	R	0x00	0x27
slt	Set to 1 if Less Than	R	0x00	0x2A
sltu	Set to 1 if Less Than Unsigned	R	0x00	0x2B
j	Jump to Address	J	0x02	NA
jal	Jump and Link	J	0x03	NA
beq	Branch if Equal	I	0x04	NA
bne	Branch if Not Equal	I	0x05	NA
blez	Branch if Less Than or Equal to Zero	I	0x06	NA
addi	Add Immediate	I	0x08	NA
addiu	Add Unsigned Immediate	I	0x09	NA
slti	Set to 1 if Less Than Immediate	I	0x0A	NA
sltiu	Set to 1 if Less Than Unsigned Immediate	I	0x0B	NA
andi	Bitwise AND Immediate	I	0x0C	NA
ori	Bitwise OR Immediate	I	0x0D	NA
xori	Bitwise XOR (Exclusive-OR) Immediate	I	0x0E	NA
lui	Load Upper Immediate	I	0x0F	NA
mfc0	Move from Coprocessor 0	R	0x10	NA
lb	Load Byte	I	0x20	NA
lh	Load Halfword	I	0x21	NA
lw	Load Word	I	0x23	NA
lbu	Load Byte Unsigned	I	0x24	NA
lhu	Load Halfword Unsigned	I	0x25	NA
sb	Store Byte	I	0x28	NA
sh	Store Halfword	I	0x29	NA
sw	Store Word	I	0x2B	NA

MNEMONIC	MEANING	TYPE	OPCODE	FUNCT
add	Add	R	0x00	0x20
addi	Add Immediate	I	0x08	NA
addiu	Add Unsigned Immediate	I	0x09	NA
addu	Add Unsigned	R	0x00	0x21
and	Bitwise AND	R	0x00	0x24
andi	Bitwise AND Immediate	I	0x0C	NA
beq	Branch if Equal	I	0x04	NA
blez	Branch if Less Than or Equal to Zero	I	0x06	NA
bne	Branch if Not Equal	I	0x05	NA
div	Divide	R	0x00	0x1A
divu	Unsigned Divide	R	0x00	0x1B
j	Jump to Address	J	0x02	NA
jal	Jump and Link	J	0x03	NA
jr	Jump to Address in Register	R	0x00	0x08
lb	Load Byte	I	0x20	NA
lbu	Load Byte Unsigned	I	0x24	NA
lh	Load Halfword	I	0x21	NA
lhu	Load Halfword Unsigned	I	0x25	NA
lui	Load Upper Immediate	I	0x0F	NA
lw	Load Word	I	0x23	NA
mfc0	Move from Coprocessor 0	R	0x10	NA
mfhi	Move from HI Register	R	0x00	0x10
mflo	Move from LO Register	R	0x00	0x12
mult	Multiply	R	0x00	0x18
multu	Unsigned Multiply	R	0x00	0x19
nor	Bitwise NOR (NOT-OR)	R	0x00	0x27
or	Bitwise OR	R	0x00	0x25
ori	Bitwise OR Immediate	I	0x0D	NA
sb	Store Byte	I	0x28	NA
sh	Store Halfword	I	0x29	NA
sll	Logical Shift Left	R	0x00	0x00
slt	Set to 1 if Less Than	R	0x00	0x2A
slti	Set to 1 if Less Than Immediate	I	0x0A	NA
sltiu	Set to 1 if Less Than Unsigned Immediate	I	0x0B	NA
sltu	Set to 1 if Less Than Unsigned	R	0x00	0x2B
sra	Arithmetic Shift Right (sign-extended)	R	0x00	0x03
srl	Logical Shift Right (0-extended)	R	0x00	0x02
sub	Subtract	R	0x00	0x22
subu	Unsigned Subtract	R	0x00	0x23
sw	Store Word	I	0x2B	NA
xor	Bitwise XOR (Exclusive-OR)	R	0x00	0x26
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 label        (branches; immediate = offset >> 2)
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

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, preceded by '0x', 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.