The National School of Artificial Intelligence
Computer Architecture
Computer Components

Assignment 02

Exercise 1:

1. Write the following strings using ASCII encoding. Write your final answers in hexadecimal.

1) hello there	2) Cool	3) boo!
4) bag o chips	5) To the rescue!	6) RISC-V

2. Show how the previous strings are stored in a byte-addressable memory starting at memory address 0x004F05BC. Clearly indicate the memory address of each byte.

Exercise 2:

The nor instruction is not part of the RISC-V instruction set because the same functionality can be implemented using existing instructions. Write a short assembly code snippet that has the following functionality: s3 = s4 NOR s5. Use as few instructions as possible. Do the same thing for NAND.

Exercise 3:

Assume that registers x5 and x6 hold the values 0x80000000 and 0xD0000000, respectively. What is the value of x30 for the following assembly codes? Is the result in x30 the desired result, or has there been an overflow?

1. Code 1:

```
        add x30, x5, x6
```

2. Code 2:

```
        sub x30, x5, x6
```

3. Code 3:

```
        add x30, x5, x6
        add x30, x30, x5
```

Exercise 4:

Assume that x5 holds the value 128_{ten}. For the instructions below, what is the range(s) of values for x6 that would result in overflow?

1. Instruction 1:

```
        add x30, x5, x6

```

2. Instruction 2:

```
        sub x30, x6, x5

```

3. Instruction 3:

```
        sub x30, x5, x6

```

Exercise 5:

Assume the following register contents:

```
        x5 = 0xAAAAAAAA, x6 = 0x12345678
```

For the register values shown above, what is the value of x7 for the following sequences of instructions?

1. First sequence:

```
    slli x7, x5, 4
    or x7, x7, x6

```

2. Second sequence:

```
    slli x7, x6, 4

```

3. Third sequence:

```
    srli x7, x5, 3
    andi x7, x7, 0xEF

```

Exercise 6:

Write RISC-V assembly code for placing the following immediates (constants) in s7. Use a minimum number of instructions.

1) 29	2) –214	3) –2999	4) 0xABCDE000
5) 0xEDCBA123	6) 0xEEEEEFAB	7) 47	8) -349
9) 5328	10) 0xBBCCD000	11) 0xFEEBC789	12) 0xCCAAB9AB

Exercise 7:

Provide a minimal set of RISC-V instructions that may be used to implement the following pseudo instruction:

```
    not x5, x6 // bit-wise invert
```

Decimal	**Character**	**Description**	**Decimal**	**Character**	**Description**
0	NUL	Null character	32	(space)	Space
1	SOH	Start of Heading	33	!	Exclamation mark
2	STX	Start of Text	34	"	Double quote
3	ETX	End of Text	35	#	Number sign
4	EOT	End of Transmission	36	$	Dollar sign
5	ENQ	Enquiry	37	%	Percent sign
6	ACK	Acknowledge	38	&	Ampersand
7	BEL	Bell	39	’	Single quote
8	BS	Backspace	40	(	Left parenthesis
9	TAB	Horizontal Tab	41	)	Right parenthesis
10	LF	Line feed	42	*	Asterisk
11	VT	Vertical Tab	43	+	Plus sign
12	FF	Form feed	44	,	Comma
13	CR	Carriage return	45	-	Hyphen
14	SO	Shift Out	46	.	Period
15	SI	Shift In	47	/	Slash
16	DLE	Data Link Escape	48	0	Digit 0
17	DC1	Device Control 1	49	1	Digit 1
18	DC2	Device Control 2	50	2	Digit 2
19	DC3	Device Control 3	51	3	Digit 3
20	DC4	Device Control 4	52	4	Digit 4
21	NAK	Negative Acknowledge	53	5	Digit 5
22	SYN	Synchronous Idle	54	6	Digit 6
23	ETB	End of Trans. Block	55	7	Digit 7
24	CAN	Cancel	56	8	Digit 8
25	EM	End of Medium	57	9	Digit 9
26	SUB	Substitute	58	:	Colon
27	ESC	Escape	59	;	Semicolon
28	FS	File Separator	60	<	Less-than sign
29	GS	Group Separator	61	=	Equals sign
30	RS	Record Separator	62	>	Greater-than sign
31	US	Unit Separator	63	?	Question mark

64	@	At sign	96	‘	Grave accent
65	A	A	97	a	a
66	B	B	98	b	b
67	C	C	99	c	c
68	D	D	100	d	d
69	E	E	101	e	e
70	F	F	102	f	f
71	G	G	103	g	g
72	H	H	104	h	h
73	I	I	105	i	i
74	J	J	106	j	j
75	K	K	107	k	k
76	L	L	108	l	l
77	M	M	109	m	m
78	N	N	110	n	n
79	O	O	111	o	o
80	P	P	112	p	p
81	Q	Q	113	q	q
82	R	R	114	r	r
83	S	S	115	s	s
84	T	T	116	t	t
85	U	U	117	u	u
86	V	V	118	v	v
87	W	W	119	w	w
88	X	X	120	x	x
89	Y	Y	121	y	y
90	Z	Z	122	z	z
91	[	Left square bracket	123	{	Left curly brace
92	\	Backslash	124	|	Vertical bar
93	]	Right square bracket	125	}	Right curly brace
94	^	Caret	126	∼	Tilde
95	_	Underscore	127	DEL	Delete

Instruction	**Format**	**Description**	**Operation**
LUI	U-type	Load Upper Immediate	rd = imm
AUIPC	U-type	Add Upper Immediate to PC	rd = pc + imm
JAL	J-type	Jump and Link	rd = pc + 4; pc = pc + imm
JALR	I-type	Jump and Link Register	rd = pc + 4; pc = (rs1 + imm) & 1
BEQ	B-type	Branch if Equal	if (rs1 == rs2) pc = pc + imm
BNE	B-type	Branch if Not Equal	if (rs1 != rs2) pc = pc + imm
BLT	B-type	Branch if <	if (rs1 < rs2) pc = pc + imm
BGE	B-type	Branch if $\geq$	if (rs1 >= rs2) pc = pc + imm
BLTU	B-type	Branch if < Unsigned	if (rs1 < rs2) pc = pc + imm
BGEU	B-type	Branch if $\geq$ Unsigned	if (rs1 >= rs2) pc = pc + imm
LB	I-type	Load Byte	rd = sign-extend(M[rs1 + imm][7:0])
LH	I-type	Load Halfword	rd = sign-extend(M[rs1 + imm][15:0])
LW	I-type	Load Word	rd = M[rs1 + imm]
LBU	I-type	Load Byte Unsigned	rd = zero-extend(M[rs1 + imm][7:0])
LHU	I-type	Load Halfword Unsigned	rd = zero-extend(M[rs1 + imm][15:0])
SB	S-type	Store Byte	M[rs1 + imm] = rs2[7:0]
SH	S-type	Store Halfword	M[rs1 + imm] = rs2[15:0]
SW	S-type	Store Word	M[rs1 + imm] = rs2
ADDI	I-type	Add Immediate	rd = rs1 + imm
SLTI	I-type	Set < Immediate	rd = (rs1 < imm)
SLTIU	I-type	Set < Immediate Unsigned	rd = (rs1 < imm)
XORI	I-type	XOR Immediate	rd = rs1 ˆ imm
ORI	I-type	OR Immediate	rd = rs1 | imm
ANDI	I-type	AND Immediate	rd = rs1 & imm
SLLI	R-type	Shift Left Logical Immediate	rd = rs1 « shamt
SRLI	R-type	Shift Right Logical Immediate	rd = rs1 » shamt
SRAI	R-type	Shift Right Arithmetic Immediate	rd = rs1 » shamt
ADD	R-type	Add	rd = rs1 + rs2
SUB	R-type	Subtract	rd = rs1 - rs2
SLL	R-type	Shift Left Logical	rd = rs1 « rs2
SLT	R-type	Set <	rd = (rs1 < rs2)

Instruction	**Format**	**Description**	**Operation**
SLTU	R-type	Set < Unsigned	rd = (rs1 < rs2)
XOR	R-type	XOR	rd = rs1 ˆ rs2
SRL	R-type	Shift Right Logical	rd = rs1 » rs2
SRA	R-type	Shift Right Arithmetic	rd = rs1 » rs2
OR	R-type	OR	rd = rs1 | rs2
AND	R-type	AND	rd = rs1 & rs2