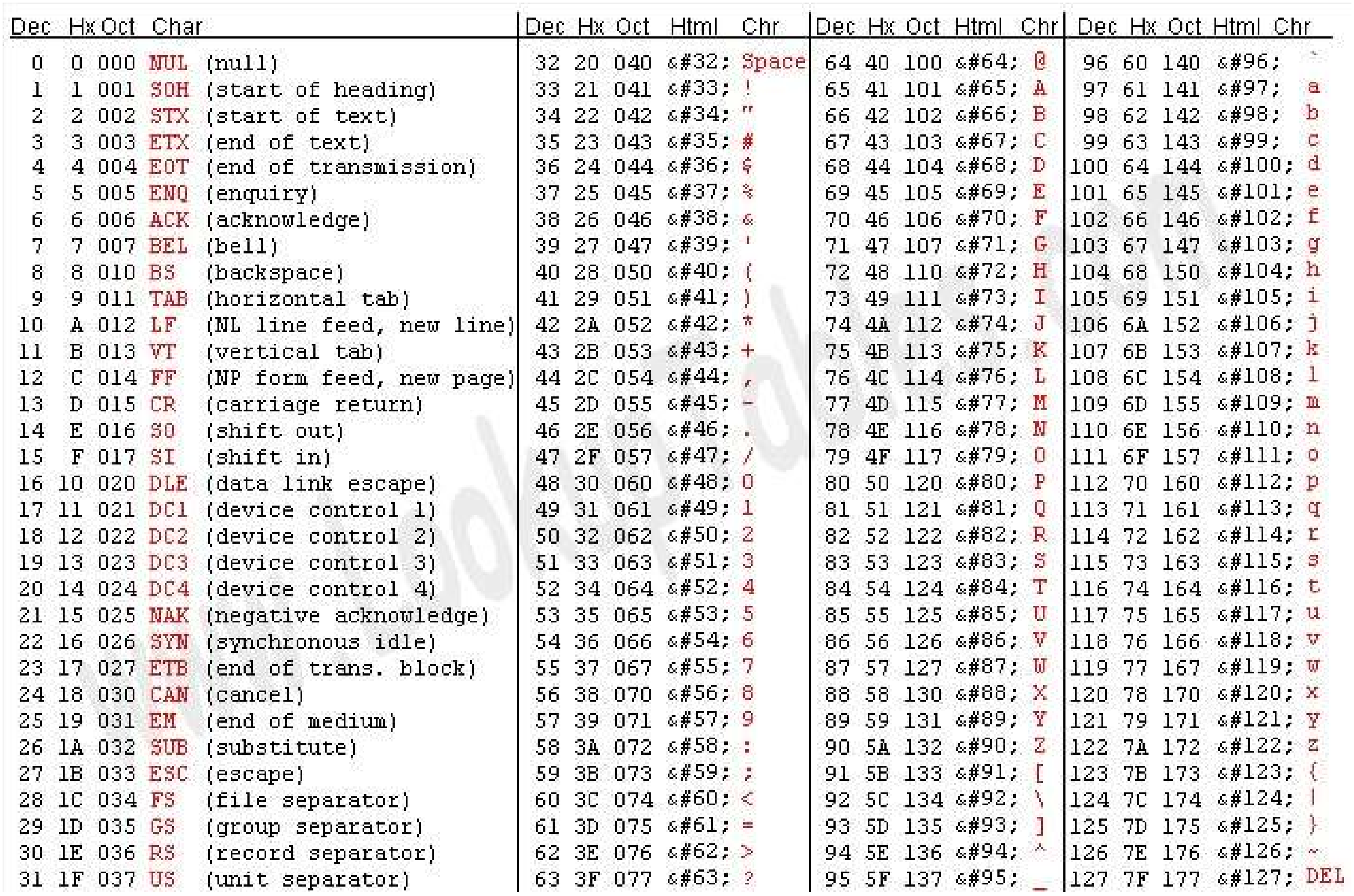
CMPUT 229

Lab #2: Caesar Cipher

American Standard Code for Information Interchange (ASCII)



Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH	(start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX	(start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX	(end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT	(end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ	(enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK	(acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL	(bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS	(backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB	(horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT	(vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR	(carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO	(shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI	(shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE	(data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1	(device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2	(device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3	(device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4	(device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK	(negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN	(synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB	(end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN	(cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM	(end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB	(substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC	(escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS	(file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS	(group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS	(record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US	(unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Strings in Assembly

Strings are arrays of characters stored in 1 byte in memory. The end of a string is indicated by a null terminator character which has a value of 0.

The string “RISC V” is represented in assembly by a pointer to the first character.

Registers	Values
t0	0x100

t0 points to the string “RISC V”

Address	0x100				0x104		
Value	R	I	S	C		V	\0

But remember, characters are stored as ASCII integer values.

Strings in Assembly

But remember, characters are actually represented by ASCII integer values

So in memory, RISC V would actually look like this:

Registers	Values
t0	0x100

t0 points to the string “RISC V”

Address	0x100				0x104		
Value	0x52	0x49	0x53	0x43	0x20	0x56	0x00

Instructions to Manipulate Characters (Bytes)

Address	0x100				0x104		
Value	R	I	S	C		V	\0
	1 byte wide	1 byte wide	…				

`lb rd, offset(rs1)`
Sign-extend to 32 bits in rd

`lbu rd, offset(rs1)`
Zero-extend to 32 bits in rd

`sb rs2, offset(rs1)`
Store just the rightmost byte of rs2

Caesar Cipher

A Caesar Cipher is a type of substitution cipher. Every character is shifted by some key.

For this lab, we will have an uppercase key and a lowercase key that should be used to encrypt a letter depending on its case.

You should leave all spaces in their original position. There will not be any punctuation in the strings.

Caesar Cipher Visualized

(check lab page if you can’t watch video^^^)

Allocating Memory

To complete this lab, you **must** allocate memory for the new encrypted string. You cannot simply overwrite the original string provided to your function.

Static Allocation: Reserving an area of memory where the size is known at compile time. In other words, you know the amount of memory you need without executing the program.

Static Allocation in RARS:

Sample Code

To access your static memory use load address:

la t0, static_space
la t1, counter

```
.data
    buffer: .space 64
    counter: .word 1

.include "common.s"

.text
    # your instructions go here
    # ...
```

Allocating Memory

In this lab, since the size of the input string is not known, static allocation is infeasible. Instead, a solution should use dynamic allocation.

Dynamic Allocation: Reserving an area of memory where the size does not have to be known. Depending on the execution of the program, you could have different memory sizes.

Dynamic Allocation in RARS:

To dynamically allocate memory, set a7 to 9 and a0 to the number of bytes in memory. The ecall instruction allocates the requested space in memory. It will return a0 as a pointer to the allocated memory

Sample Code

```
li a7, 9 # a7 <- 9
addi a0, x0, 64 # a0 <- 64
ecall
# now a0 stores a pointer to 64 bytes of contiguous memory

# … instructions that use memory
```

Modulo in RISC-V

rem rd, rs1, rs2

- Stores the remainder of rs1/rs2 into rd

rem t0, t1, t2

- t0 = t1 % t2

t0 = 9 % 4
t0 = 1

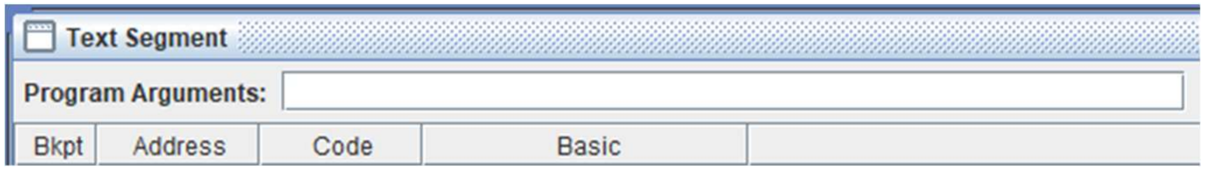
Registers	Values
t0	1
t1	9
t2	4

Testing your Lab

We have provided some test inputs and outputs for you to confirm that your lab is working.

In the Program Arguments bar in the Execute tab, enter the complete path to the test file. If your path has any spaces in it, the filename will not be read correctly by RARS.

For example, UofAstudent/cmput 229/lab 2/test3.txt is an invalid path.



caesarEncrypt

Parameter:

a0: pointer to a string to encrypt

a1: uppercase key

a2: lowercase key

Return Value:

a0: pointer to a newly allocated memory that contains the encrypted string.

What to Submit?

A single file, called **caesarencrypt.s**.

Make sure the file **does not** contain a main procedure.