Computer Architecture HW1

TA: 梁瀚中

Due: Nov. 8, 2021 (11 p.m.)

Email: r10943006@ntu.edu.tw

Outline

- Jupiter: RISC-V Simulator
- HW1-1 Recursive Function
- HW1-2 Encryption
- Report
- Rules
- Submission

Jupiter: RISC-V Simulator

- An open source RISC-V assembler and runtime simulator
- Download here: https://github.com/andrescv/Jupiter

Installation

Download the app image for your operating system and unzip the file:

- Jupiter v3.1 - Linux (Ubuntu)
- Jupiter v3.1 - macOS
- Jupiter v3.1 - Windows

Running Jupiter on Linux or macOS

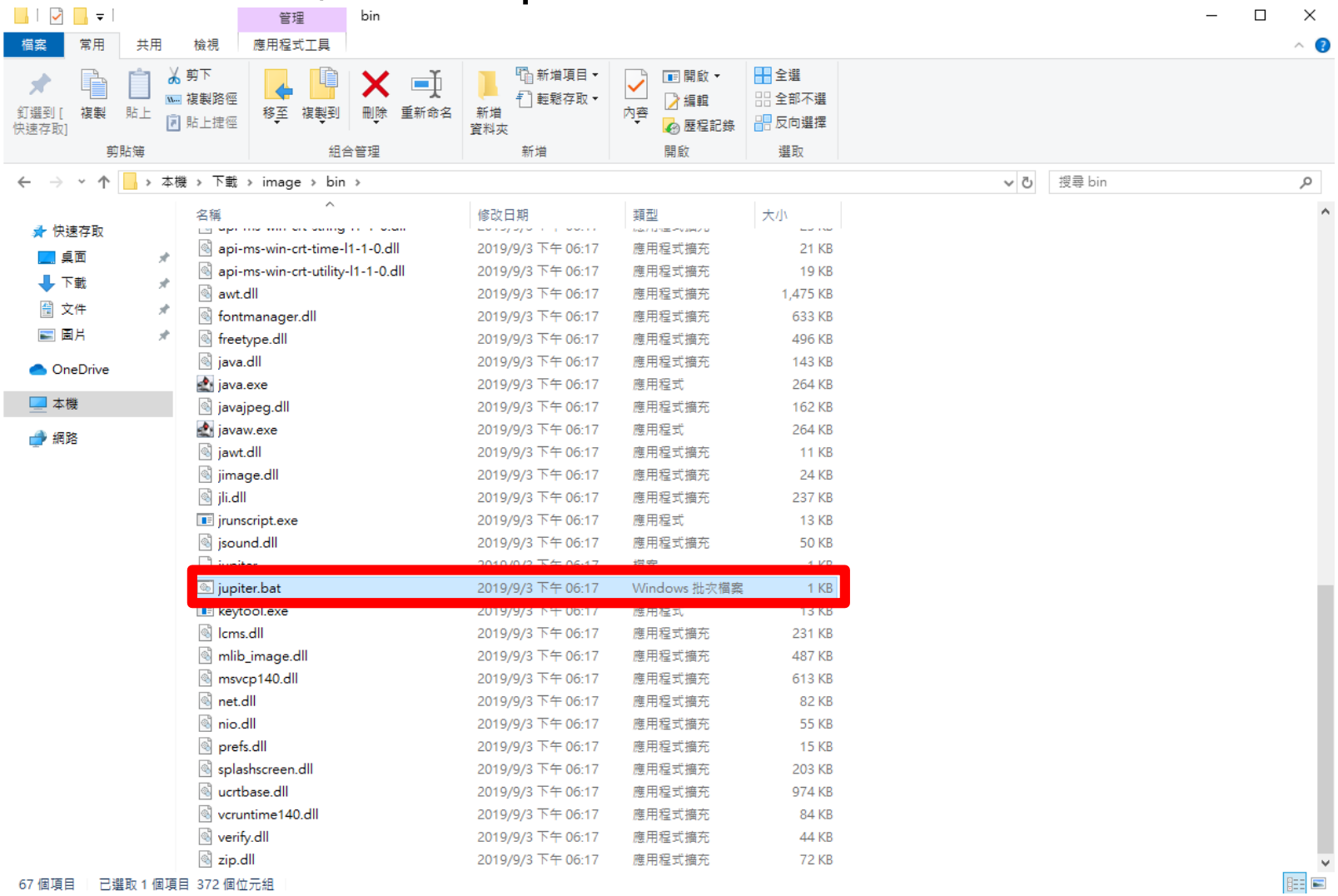
```
./image/bin/jupiter # for GUI mode
./image/bin/jupiter [options] <files> # for CLI mode
```

Running Jupiter on Windows

```
image\bin\jupiter # for GUI mode
image\bin\jupiter [options] <files> # for CLI mode
```

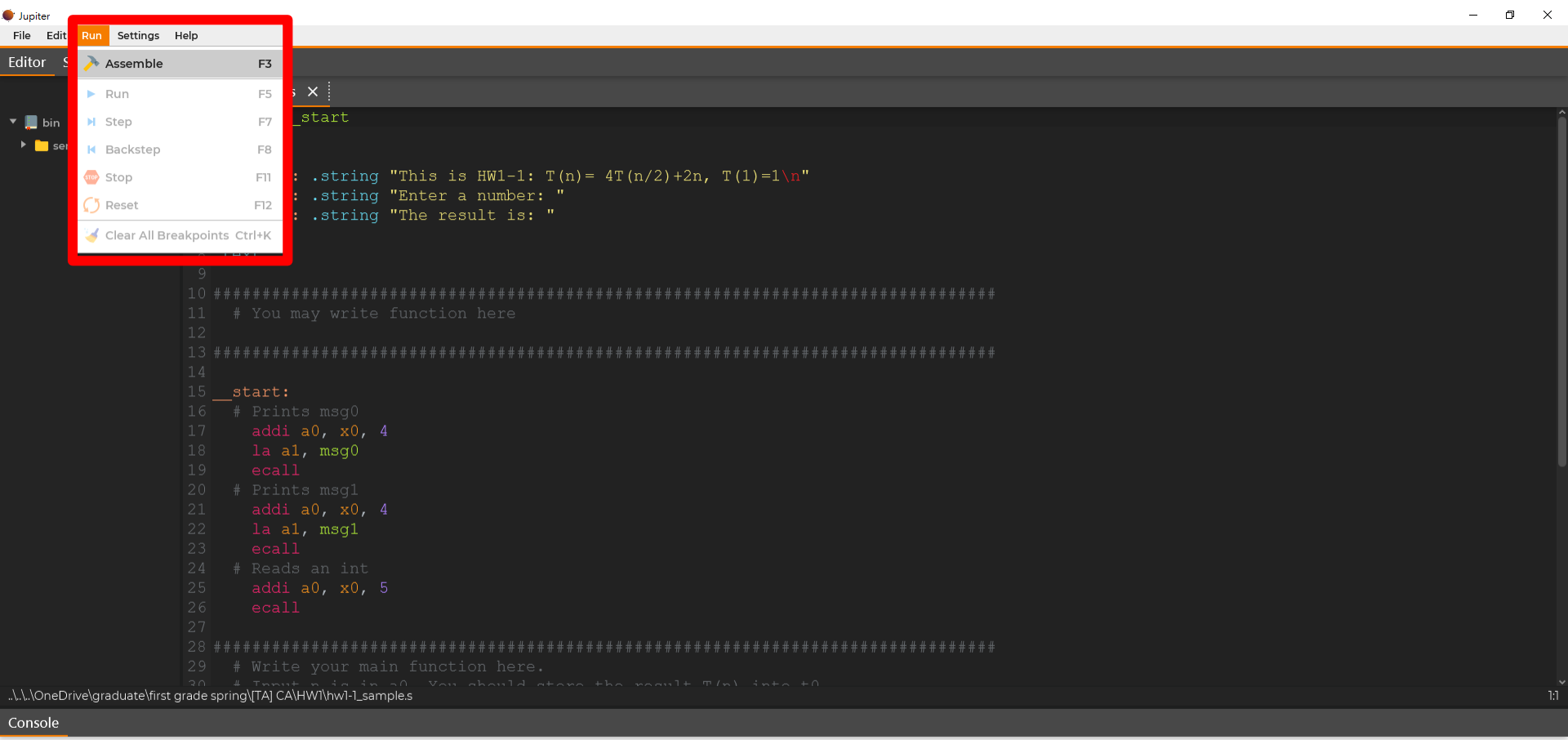
Jupiter: RISC-V Simulator (Cont.)

- To launch it, click Jupiter.bat in bin



GUI of Jupiter

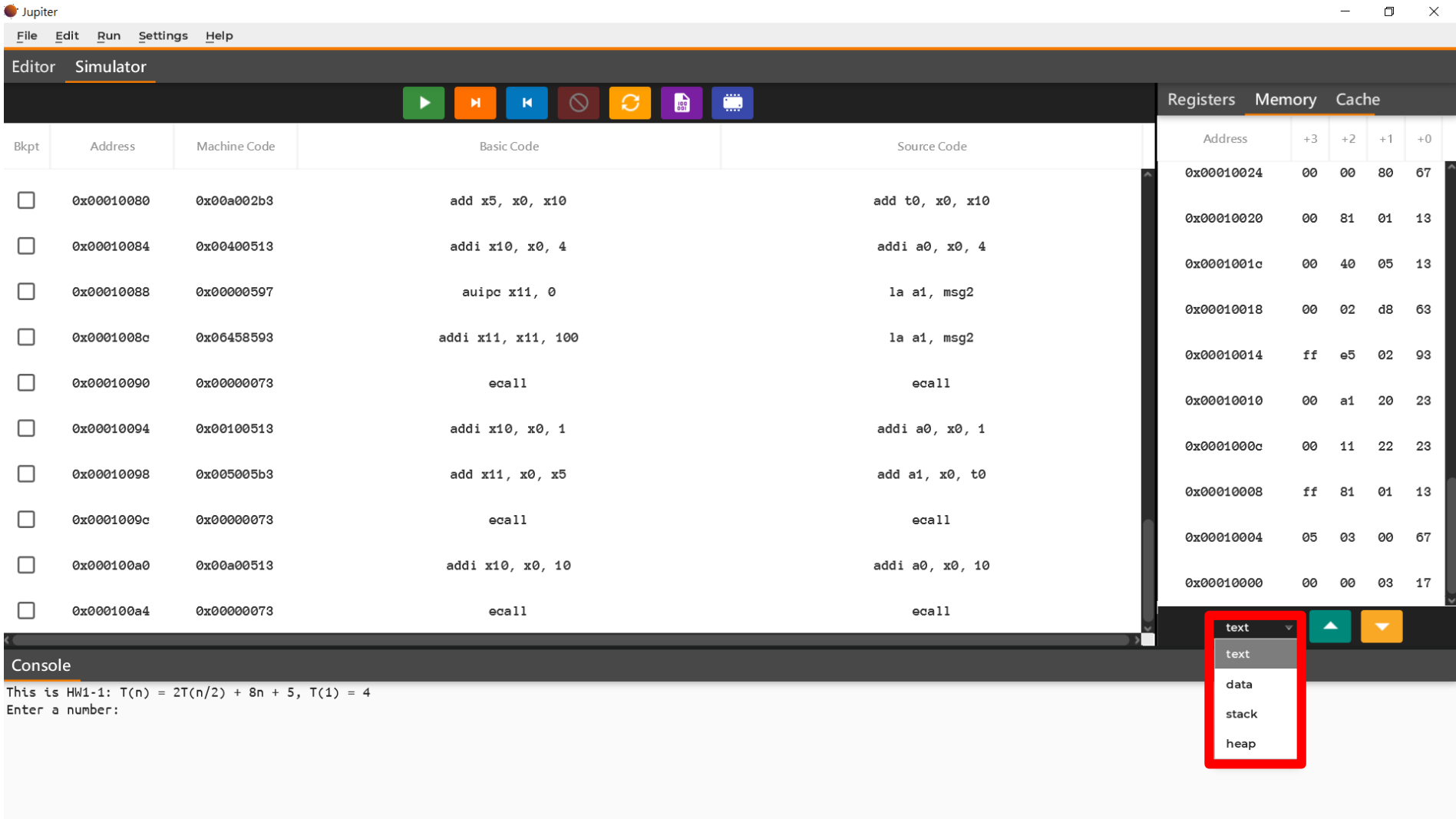
- To run the code
  - Click Run -> Assemble



GUI of Jupiter (Cont.)

GUI of Jupiter (Cont.)

- Memory type: text, data, stack, heap



HW1-1 Recursive Function

- Input
  - A positive integer n
- Output $T(n)$
  - $T(n) = \begin{cases} 2T\left(\lfloor \frac{n}{2} \rfloor\right) + 8n + 5, & if\ n \geq 2 \\ 4, & n = 1 \end{cases}$
  - e.g., $T(10) = 291$, $T(30) = 1035$
- Round down the result of division to an integer
  - e.g., 3/2 = 1, 7/3 = 2
- Implement with recursive function only

Template of Homework 1-1

- The input stores in a0 (i.e., x10)
- The output should be stored into t0 (i.e., x5)
- Write your code in the red frame
  - You may use a function and write a jump to execute it.

```
.globl __start

.rodata
    msg0: .string "This is HW1-1: T(n) = 8T(n/2) + 4n, T(1) = 7\n"
    msg1: .string "Enter a number: "
    msg2: .string "The result is: "

.text


__start:
  # Prints msg0
    addi a0, x0, 4
    la a1, msg0
    ecall

  # Prints msg1
    addi a0, x0, 4
    la a1, msg1
    ecall

  # Reads an int
    addi a0, x0, 5
    ecall

  ###############################################################################
    # Write your main function here.
    # Input n is in a0. You should store the result T(n) into t0
    # HW1-1 T(n) = 8T(n/2) + 4n, T(1) = 7, round down the result of division
    # ex. addi t0, a0, 1

  ###############################################################################

result:
  # Prints msg2
    addi a0, x0, 4
    la a1, msg2
    ecall

  # Prints the result in t0
    addi a0, x0, 1
    add a1, x0, t0
    ecall

  # Ends the program with status code 0
```

You May Ask

- What is a0 and t0
- It is just a mnemonic
- In this homework, you can use any registers you want

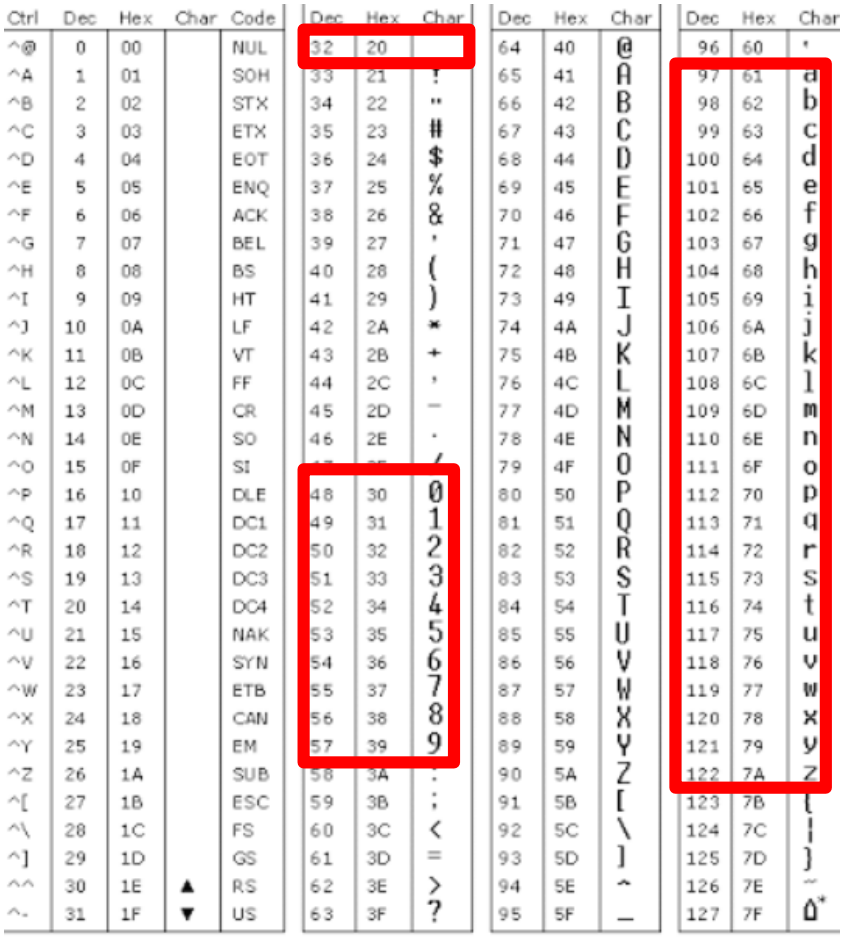
Mnemonic	Number	Value
gp	x3	0x10008000
tp	x4	0x00000000
t0	x5	0x00000000
t1	x6	0x00000000
t2	x7	0x00000000
s0	x8	0x00000000
s1	x9	0x00000000
a0	x10	0x00000000
a1	x11	0x00000000
a2	x12	0x00000000
a3	x13	0x00000000
a4	x14	0x00000000
a5	x15	0x00000000
a6	x16	0x00000000

HW1-2 Encryption

- a-z: use Caesar cipher
  - Plaintext: abcdefghijklmnopqrstuvwxyz
  - Ciphertext: defghijklmnopqrstuvwxyzabc
- Space: encode to incremental integers starting from 0
  - Plaintext is “abc and cde”
  - Ciphertext is “def0dqg1fgh”
- Input
  - Inputs are only lower-case alphabets and spaces
  - The count of spaces will not exceed ten
- Output
  - You must store the ciphertext in memory address from 66048(0x10200)
- Use “j print_char” when your code is finished

HW1-2 Encryption (Cont.)

- Character are stored as ascii code
- A character is 8 bits



Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01		SOH
^B	2	02		STX
^C	3	03		ETX
^D	4	04		EOT
^E	5	05		ENQ
^F	6	06		ACK
^G	7	07		BEL
^H	8	08		BS
^I	9	09		HT
^J	10	0A		LF
^K	11	0B		VT
^L	12	0C		FF
^M	13	0D		CR
^N	14	0E		SO
^O	15	0F		SI
^P	16	10		DLE
^Q	17	11		DC1
^R	18	12		DC2
^S	19	13		DC3
^T	20	14		DC4
^U	21	15		NAK
^V	22	16		SYN
^W	23	17		ETB
^X	24	18		CAN
^Y	25	19		EM
^Z	26	1A		SUB
^[	27	1B		ESC
^\	28	1C		FS
^]	29	1D		GS
^^	30	1E	▲	RS
^-	31	1F	▼	US

Dec	Hex	Char
32	20	
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂

HW1-2 Encryption (Cont.)

- The function “print_char” have been provided in the sample
- Usage:
  - 1. Store the beginning address in x20
  - 2. Use "j print_char"
  - The function will print the string stores from x20
  - When finished, the whole program with return value 0

Template of Homework 1-2

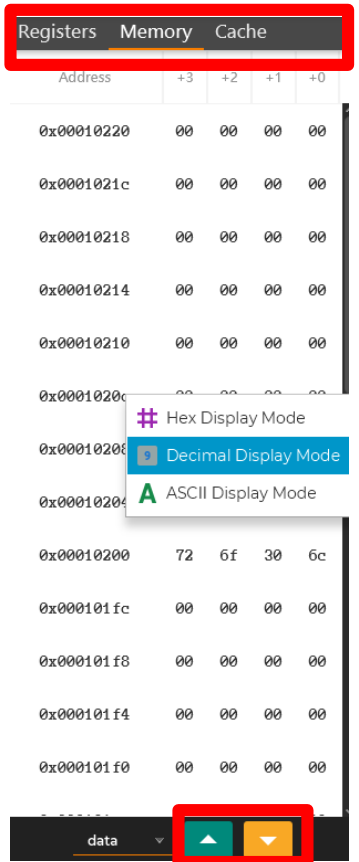
- The plaintext stores in a0 (i.e., x10)
- Do store “66048(0x10200)” in x20 before jump to print_char
- Write your code in the red frame

```
9   ###############################################################################
10    # print_char function
11    # Usage:
12    #     1. Store the beginning address in x20
13    #     2. Use "j print_char"
14    #     The function will print the string stored from x20
15    #     When finish, the whole program with return value 0
16
17  print_char:
18      addi a0, x0, 4
19      la a1, msg2
20      ecall
21
22      add a1,x0,x20
23      ecall
24
25    # Ends the program with status code 0
26      addi a0,x0,10
27      ecall
28
29  ###############################################################################
```

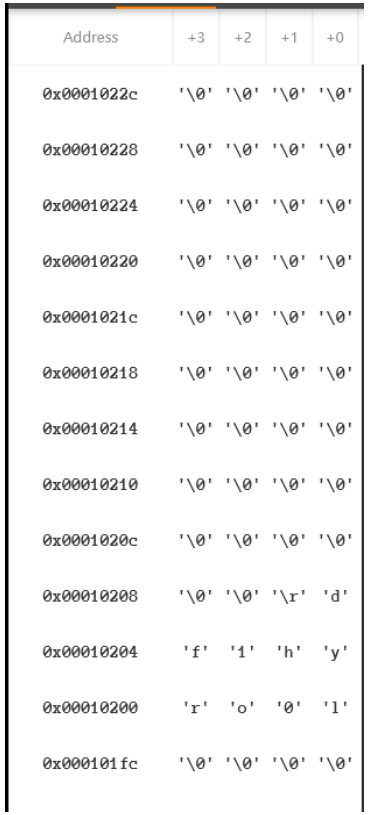
```
29  ###############################################################################
30
31  __start:
32    # Prints msg
33      addi a0, x0, 4
34      la a1, msg0
35      ecall
36
37      la a1, msg1
38      ecall
39
40      addi a0,x0,8
41      li a1, 0x10130
42      addi a2,x0,2047
43      ecall
44
45    # Load address of the input string into a0
46      add a0,x0,a1
47
48  ###############################################################################
49    # Write your main function here.
50    # a0 stores the begining Plaintext
51    # Do store 66048(0x10200) into x20
52    # ex. j print_char
53
54  ###############################################################################
```

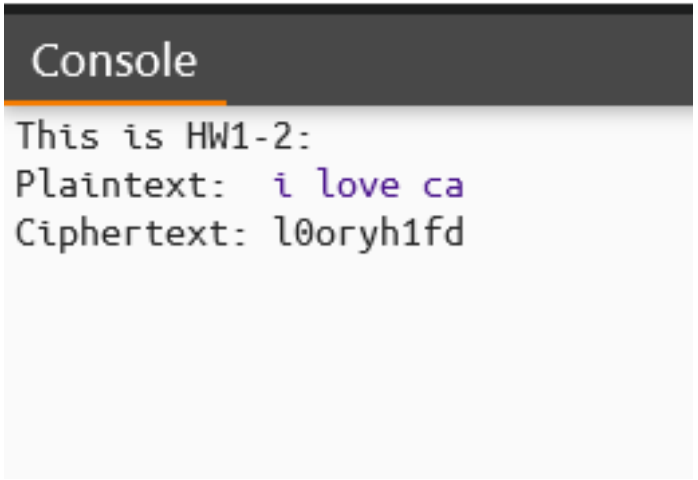
HW1 Report

- HW1-1: snapshot the result with the input n=10
- HW1-2: snapshot the result with the plaintext = i love ca and the value in memory 0x10200 as ascii code format
- Make this file into a pdf file (read the submission)



Right click and choose ASCII





Rules

- For HW 1-1 and 1-2, brute-force is not allowed
  - Implement HW1-1 with recursive function
  - Implement HW1-2 with loop function only
- Please do write some comments in your codes
- Input will be changed while grading
- Do NOT modify the input, output, and any provided instructions

Submission

- Deadline: Nov. 8, 2021 (11 p.m.)
  - No late submission allowed
- Hand in two source codes and a pdf report on ceiba
- Your homework should be copied into a folder and packed into a zip file with the following naming rules
  - hw1_<student_id>.zip
    - hw1_<student_id>
      - hw1-1_<student_id>.s
      - hw1-2_<student_id>.s
      - hw1_report _<student_id>.pdf
  - Ex: hw1_r10943006.zip