ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ

o0o

***

BÁO CÁO

BÀI TẬP LỚN CẤU TRÚC MÁY TÍNH

Đề tài:
Thiết kế CPU chạy đơn chu kỳ và Pipeline dựa trên cấu trúc CPU RV32

GVHD:Thầy Trần Hoàng Linh
SVTT:Dương Nhật Huy – 181016

1. Thiết kế CPU RV32

a) Phân tích

CPU RV32 có tổng cộng 32 lệnh hợp ngữ, trong đó mỗi lệnh có độ dài 32bit và có 7 bit [6:0] (opcode) để xác định loại lệnh.

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND

- Tập lệnh của RV32 còn được gọi là tập lệnh kiểu load-store, điều đó có nghĩa là data trong bộ nhớ muốn được thực thi thì trước hết phải được lấy ra bỏ vào băng thanh ghi rồi mới được tính toán. Sau khi tính toán, data sẽ được lưu lại vào memory.

- Các thanh ghi trong băng thanh ghi (Register Bank) có độ dài 32 bits và có 32 thanh ghi (từ xo − x31) ⟹ Cần có 5 bits để xác định địa chỉ của các thanh ghi trong băng thanh ghi. Trong đó, chức năng của 32 thanh ghi được cho như ở bảng dưới.

Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5	t0	Temporary/alternate link register	Caller
x6–7	t1–2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10–11	a0–1	Function arguments/return values	Caller
x12–17	a2–7	Function arguments	Caller
x18–27	s2–11	Saved registers	Callee
x28–31	t3–6	Temporaries	Caller

Thanh ghi xo luôn có giá trị bằng 0x00000000 và không thay đổi giá trị này.

- Nhóm lệnh R-Format:

Nhóm lệnh này bao gồm các lệnh có cấu trúc như ở hình sau:

0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND

Nhóm lệnh này có opcode là [6:0] = 0110011

Nhóm lệnh này thực hiện lấy hai giá trị lưu ở thanh ghi rs1 và rs2 thực hiện đưa vào khối ALU để tính toán, sau đó lưu kết quả vào thanh ghi rd.

- Nhóm lệnh I (Tính toán):

imm[11:0]		rs1	000	rd	0010011	addi
imm[11:0]		rs1	010	rd	0010011	slti
imm[11:0]		rs1	011	rd	0010011	sltiu
imm[11:0]		rs1	100	rd	0010011	xori
imm[11:0]		rs1	110	rd	0010011	ori
imm[11:0]		rs1	111	rd	0010011	andi
0000000	shamt	rs1	001	rd	0010011	slli
0000000	shamt	rs1	101	rd	0010011	srli
0100000	shamt	rs1	101	rd	0010011	srai

Nhóm lệnh này có opcode là [6:0] = 0010011

Nhóm lệnh này (trừ 3 lệnh SRAI, SRLI, SLLI) thực hiện lấy giá trị lưu ở thanh ghi rs1 và giá trị lưu ở imm[11:0] (được mở rộng dấu), thực hiện đưa vào khối ALU để tính toán. Kết quả được lưu vào thanh ghi rd.

- Nhóm lệnh I (Load data)

imm[11:0]	rs1	000	rd	0000011	lb
imm[11:0]	rs1	010	rd	0000011	lh
imm[11:0]	rs1	011	rd	0000011	lw
imm[11:0]	rs1	100	rd	0000011	lbu
imm[11:0]	rs1	110	rd	0000011	lhu

Nhóm lệnh này có opcode là [6:0] = 0000011

Nhóm lệnh này thực hiện lấy hai giá trị lưu ở thanh ghi rs1 và giá trị lưu ở imm[11:0] (được mở rộng dấu) để tính tổng rs1 + ext(imm[11:0]). Sau đó lấy giá trị lưu trong data memory tại địa chỉ rs1 + ext(imm[11:0]).Với lb,lh,lw lần lượt lấy 8 , 16 , 32 bit của giá trị này lưu vào rd. Do lb và lh chỉ lấy 8 , 16 bit nên các bit trống của rd được lấp đầy bằng dấu của giá trị lấy ra từ data memory

- Nhóm lệnh S (Store data)

Imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	sb
Imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	sh
Imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	sw

Nhóm lệnh này có opcode là [6:0] = 0100011

Nhóm lệnh này thực hiện lấy hai giá trị lưu ở thanh ghi rs1 và giá trị lưu ở imm[11:5] và imm[4:0] (ghép lại và mở rộng dấu) để tính tổng rs1 + ext(imm[11:5]imm[4:0]). Sau đó lấy giá trị lưu trong thanh ghi rs2 . Với các lệnh sb , sh , sw lưu 8 , 16 ,32 bit thấp của giá trị này vào DMEM tại địa chỉ rs1 + ext(imm[11:5]imm[4:0]).

- Nhóm lệnh B (rẽ nhánh)

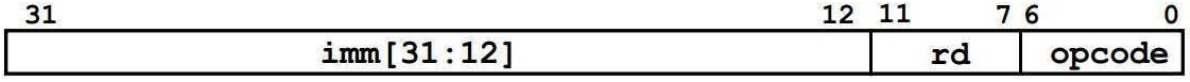
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU

Nhóm lệnh này có opcode là [6:0] = 1100011

Nhóm lệnh này sẽ thực hiện chuyển giá trị của thanh ghi PC thành giá trị được lưu trong các phần imm giá trị lưu trong rs1 và rs2 thỏa điều kiện câu lệnh (bằng, không bằng, nhỏ hơn ,lớn hơn, nhỏ hơn không dấu, lớn hơn không dấu)

Khi lấy giá trị lưu ở phần imm ta phải ghép lại cho đúng thứ tự và mở rộng dấu, bit LSB luôn luôn bằng 0.

- Nhóm lệnh U



+ Với lệnh LUI :

Opcode = 0110111. Lệnh này load giá trị imm[31:12]000000000000 vào thanh ghi rd.

+ Với lệnh AUIPC

Opcode = 0010111. Lệnh này load giá trị ở PC vào thanh ghi rd.

- Nhóm lệnh J (nhảy không điều kiện)

imm[20|10:1|11|19:12]			rd	1101111	JAL
imm[11:0]	rs1	000	rd	1100111	JALR

+ Với lệnh JAL :

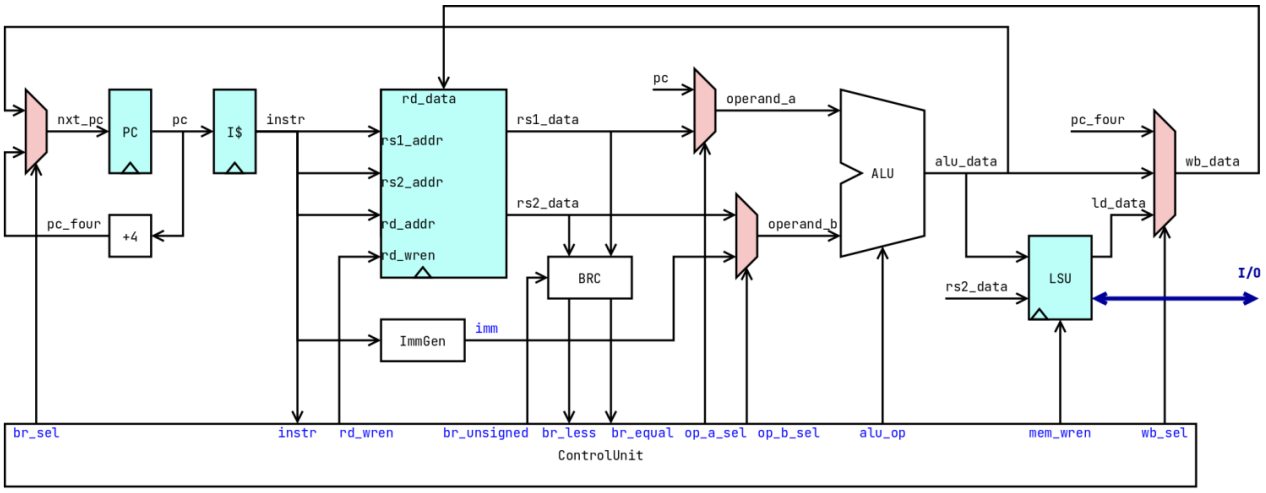
x[rd] = pc+4; pc = pc + ext(imm);

+ Với lệnh JALR:

x[rd] = pc+4; pc =(x[rs1]+ext(imm)) & ~1;

b) Thiết kế CPU RV32

i) Sơ đồ khối đơn chu kỳ:



ii) Mô tả

- Inputs:
  + clk_i, rst_ni: xung clock và tín hiệu reset dữ liệu.
  + take_bit: quyết định thực hiện lệnh nào trong các lệnh lw, lh, lb, lbu, lbhu, sw, sb, sh
  + addr: 32 bit địa chỉ đọc hoặc ghi dữ liệu
  + st_data: 32 bit dữ liệu cần lưu giữ trong LSU
  + st_en: 1 nếu ghi dữ liệu, 0 đọc dữ liệu.
  + io_sw: 18-bit data từ 18 switches trên Kit DE2
- Output:
  + ld_data: 32 bit load data.
  + io_lcd: 32 bit data to drive LCD.
  + io_ledg: 32-bit data to drive green LEDs.
  + io_ledr: 32-bit data to drive red LEDs.
  + io_hex0 .. io_hex7: 8 32-bit data to drive 8 7-segment LEDs.

- Tiến hành lập bảng bao gồm các lệnh, các tín hiệu vào, tín hiệu ra và phân tích câu lệnh:

1									0 => PC =PC+4	000 => Inst[31:20], ext sign	0=> ReadOnly	0 => sign	0 => B	0 => A	0000 => Add	0 => Read	00 => W	000 => W	00 => DMEM
2									1 => PC from ALU	001 => Inst[31:20], ext usign	1 => Write	1 => usign	1 => imm	1 => PC	0001 => Sub	1 => Write	01 => B	001 => W -> extB	01 => ALU
3										010 => Inst[24:20], ext usign					0010 => ShiftLeft		10 => HW	010 => W -> extHW	10 => PC +4
4										011 => Inst[31:25][11:7], ext sign					…			101 => W -> extUB	
5										100 => Inst[31][7][30:25][11:8]0, ext sign								110 => W -> extUHW	
6										101 => Inst[31:12]0..0									
7										110 => Inst[31][19:12][20][30:21]0, ext sign									
8																			
9	No	Type	Name	Inst[30]	Inst[14:12]	Inst[6:2]	BrEq	BrLT	PCSel	ImmSel	RegWEn	BrUN	Bsel	Asel	ALUSel	MemRW	DataIn	DataOutAddj	WBSel
10	1	R	ADD	0	000	01100	x	x	0	x	1	x	0	0	0000	0	x	x	01
11	2	R	SUB	1	000	01100	x	x	0	x	1	x	0	0	0001	0	x	x	01
12	3	R	SLL	0	001	01100	x	x	0	x	1	x	0	0	0010	0	x	x	01
13	4	R	SLT	0	010	01100	x	x	0	x	1	x	0	0	0011	0	x	x	01
14	5	R	SLTU	0	011	01100	x	x	0	x	1	x	0	0	0100	0	x	x	01
15	6	R	XOR	0	100	01100	x	x	0	x	1	x	0	0	0101	0	x	x	01
16	7	R	SRL	0	101	01100	x	x	0	x	1	x	0	0	0110	0	x	x	01
17	8	R	SRA	1	101	01100	x	x	0	x	1	x	0	0	0111	0	x	x	01
18	9	R	OR	0	110	01100	x	x	0	x	1	x	0	0	1000	0	x	x	01
19	10	R	AND	0	111	01100	x	x	0	x	1	x	0	0	1001	0	x	x	01
20																			
21	11	I	ADDI	x	000	00100	x	x	0	000	1	x	1	0	0000	0	x	x	01
22	12	I	SLTI	x	010	00100	x	x	0	000	1	x	1	0	0011	0	x	x	01
23	13	I	SLTIU	x	011	00100	x	x	0	001	1	x	1	0	0100	0	x	x	01
24	14	I	XORI	x	100	00100	x	x	0	000	1	x	1	0	0101	0	x	x	01
25	15	I	ORI	x	110	00100	x	x	0	000	1	x	1	0	1000	0	x	x	01
26	16	I	ANDI	x	111	00100	x	x	0	000	1	x	1	0	1001	0	x	x	01
27	17	I	SLLI	0	001	00100	x	x	0	010	1	x	1	0	0010	0	x	x	01
28	18	I	SRLI	0	101	00100	x	x	0	010	1	x	1	0	0110	0	x	x	01
29	19	I	SRAI	1	101	00100	x	x	0	010	1	x	1	0	0111	0	x	x	01
30																			
31	20	L	LB	x	000	00000	x	x	0	000	1	x	1	0	0000	0	x	001	00
32	21	L	LH	x	001	00000	x	x	0	000	1	x	1	0	0000	0	x	010	00
33	22	L	LW	x	010	00000	x	x	0	000	1	x	1	0	0000	0	x	000	00
34	23	L	LBU	x	100	00000	x	x	0	000	1	x	1	0	0000	0	x	101	00
35	24	L	LHU	x	101	00000	x	x	0	000	1	x	1	0	0000	0	x	110	00

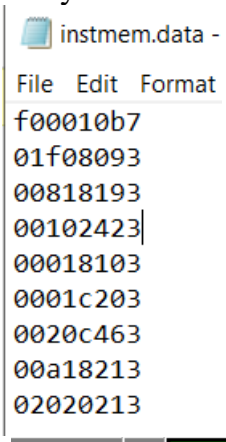
37	25	S	SB	x	000	01000	x	x	0	011	0	x	1	0	0000	1	01	x	x		
38	26	S	SH	x	001	01000	x	x	0	011	0	x	1	0	0000	1	10	x	x		
39	27	S	SW	x	010	01000	x	x	0	011	0	x	1	0	0000	1	00	x	x		
40																					
41	28	B	BEQ	x	000	11000	1	x	1	100	0	x	1	1	0000	0	x	x	x		
42							0		0												
43	29	B	BNE	x	001	11000	0	x	1	100	0	x	1	1	0000	0	x	x	x		
44							0		0												
45	30	B	BLT	x	100	11000	x	1	1	100	0	0	1	1	0000	0	x	x	x		
46								0	0												
47	31	B	BGE	x	101	11000	x	0	1	100	0	0	1	1	0000	0	x	x	x		
48								1	0												
49	32	B	BLTU	x	110	11000	x	1	1	100	0	1	1	1	0000	0	x	x	x		
50								0	0												
51	33	B	BGEU	x	111	11000	x	0	1	100	0	1	1	1	0000	0	x	x	x		
52								1	0												
53																					
54	34	U	LUI	x	x	01101	x	x	0	101	1	x	1	x	1010	0	x	x	01		
55	35	U	AUIPC	x	x	00101	x	x	0	101	1	x	1	1	0000	0	x	x	01		
56																					
57	36	J	JAL	x	x	11011	x	x	1	110	1	x	1	1	0000	0	x	x	10		
58	37	J	JALR	x	x	11001	x	x	1	000	1	x	1	0	0000	0	x	x	10		

c) Kiểm tra thiết kế

Sử dụng phần mềm thực hiện các lệnh trong risc – v.

```
lui  x1, 983041
addi x1, x1,31
addi x3, x3, 8
sw   x1, 8(x0)
lb   x2, 0(x3)
lbu  x4, 0(x3)
blt  x1,x2,LABLE
addi x4,x3,10
LABLE: addi x4,x4,32
```

Chuyển đổi các câu lệnh sang mã thập lục phân và lưu vào file instdata.mem



- Câu lệnh 1:
  + pc = 0
  + x1 = f0001000h
- Câu lệnh 2:
  + pc = 4h

+ x1 = x1 + 31 = f000101fh

- Câu lệnh 3:

+ pc = 8h

+ x3 = x3 + 8 = 0 +8 = 8

- Câu lệnh 4:

+ pc = ch

+ Lưu giá trị x1 = f000101fh vào địa chỉ x0 + 8h = 8h trong datamem

Câu lệnh 5:

+ pc = 10h

+ Vào datamem tại địa chỉ x3 + 0 = 8h , lấy 8 bit lưu vào x2. Do dùng lệnh lb và giá trị f0001000h là một số âm nếu là số có dấu nên các bit trống x2 điền giá trị 1.

X2 = ffffff1fh

Câu lệnh 6:

+ pc = 14h

+ Vào datamem tại địa chỉ x3 + 0 = 8h , lấy 8 bit lưu vào x2. Do dùng lệnh lbu nên các bit trống x4 điền giá trị 0 . x4 = 1fh

Câu lệnh 7:

+ pc = 18h

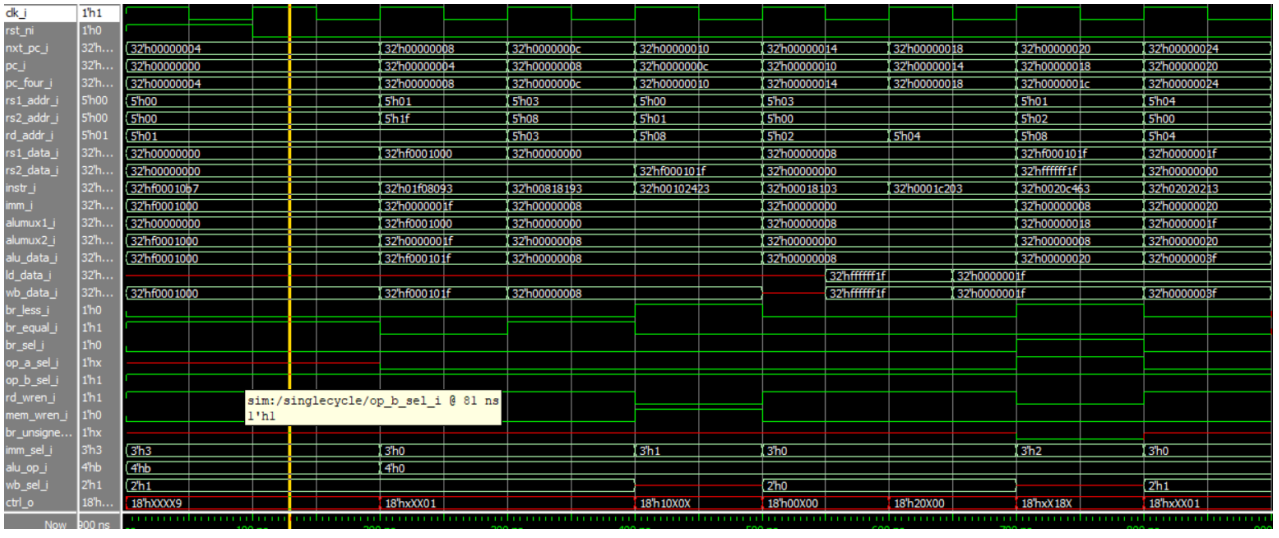
+ Nếu x1<x2 (có dấu) thì nhảy đến chỗ LABLE . Do x1 = f000101f < ffffff1f = x2 nếu so sánh có dấu nên nhảy qua câu lệnh 8 đến câu lệnh 9 (pc = 20h)

Câu lênh 9 :

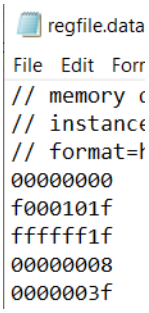
+ pc = 20h

+ x4 = x4 + 32 = 3fh

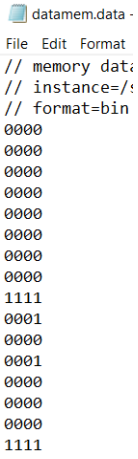
d) Kết quả mô phỏng



File regfile.data



File datamem.data



e) Kết luận

Thiết kế chạy đúng với bài test đặt ra

2. Câu hỏi

a.Write a code to find the largest number of the Fibonacci sequence less than 32'd500. You will put the ceiling number in register x10 and save the value you find in 0x400

addi x1 , x0 , 1
addi x2 , x0 , 0x0500
addi x4, x0, 1

LABLE:
add x1 , x1 , x4
add x4 , x1, x0
addi x10 , x10, 1
blt x1 , x2 , LABLE

sw x4 , 0x0400 (x0)

b. Write a code in which you read an 8-bit binary in the address 0x500, convert it into 3 decimal digits,and then write them into 0x420, 0x410, and 0x400, respectively. For example, the value in 0x500 is 0xEE, which is 238, so "2" will be in 0x420, "3" in 0x410, and "8" in 0x400

```
lbu x1 , 0x500(x0)
addi x1 , x1 , 0xEE
addi x2 , x0 , 0
addi x3 , x0 , 1
addi x4 , x0 , 100

LABLE0:
addi x2 , x2 , 100
addi x5 , x5 , 1
blt x2 , x1 , LABLE0

sub x5,x5,x3
sub x2,x2,x4
sub x1,x1,x2
addi x2, x0, 0
addi x4, x0,10
LABLE1:
addi x2 , x2 , 10
addi x6, x6, 1
blt x2 , x1 , LABLE1
sub x2,x2,x4
sub x6,x6,x3
sub x7,x1,x2

sb x5,0x420(x0)
sb x6,0x410(x0)
sb x7,0x400(x0)
```