



Homework I 說明

Instructor : Lih-Yih Chiou

TA : David

Date : 09/15/2021



Outline

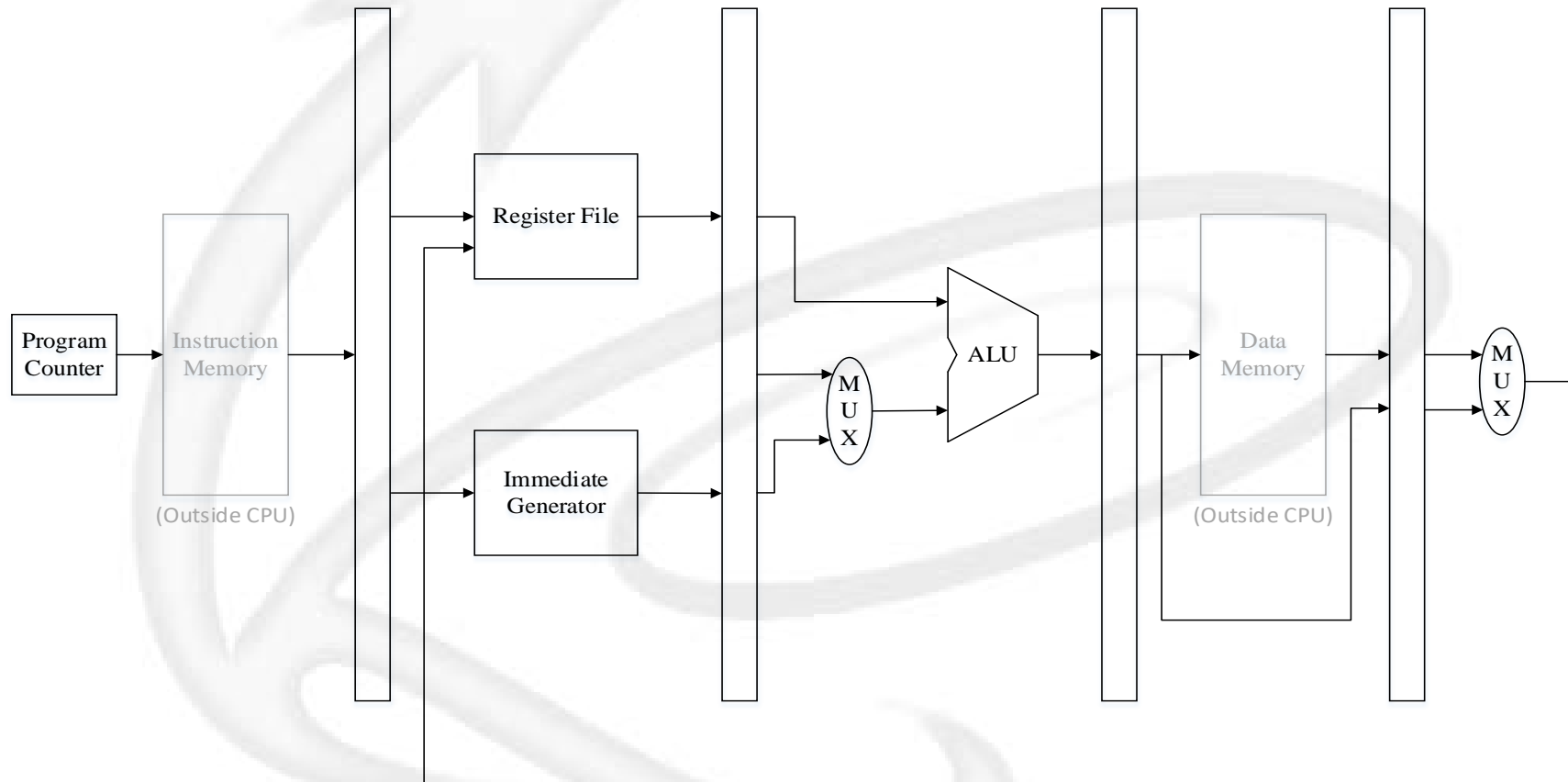
- 作業內容說明
- 作業驗證說明
- 作業繳交注意事項



作業内容説明

Problem 1

- 根據作業中附的RISC-V ISA，完成一個pipelined CPU



Problem 1 Specification

- ❑ Implement the 37 instructions as listed. The number of pipeline stage is 5.
- ❑ Register File size: 32x32-bit
→ x0 is read only 0.
- ❑ Instruction memory size: 16Kx32-bit
- ❑ Data memory size: 16Kx32-bit
- ❑ Timescale: 1ns/10ps
- ❑ Maximum Clock period: 20ns (50MHz)

Problem 1 – Instructions (1/2)

R-type

31	25	24	20	19	15	14	12	11	7	6	0		
funct7		rs2		rs1		funct3		rd		opcode		Mnemonic	Description
0000000		rs2		rs1		000		rd		0110011		ADD	rd = rs1 + rs2
0100000		rs2		rs1		000		rd		0110011		SUB	rd = rs1 - rs2
0000000		rs2		rs1		001		rd		0110011		SLL	rd = rs1 _u << rs2[4:0]
0000000		rs2		rs1		010		rd		0110011		SLT	rd = (rs1 _s < rs2 _s)? 1:0
0000000		rs2		rs1		011		rd		0110011		SLTU	rd = (rs1 _u < rs2 _u)? 1:0
0000000		rs2		rs1		100		rd		0110011		XOR	rd = rs1 _u ^ rs2
0000000		rs2		rs1		101		rd		0110011		SRL	rd = rs1 _u >> rs2[4:0]
0100000		rs2		rs1		101		rd		0110011		SRA	rd = rs1 _s >> rs2[4:0]
0000000		rs2		rs1		110		rd		0110011		OR	rd = rs1 rs2
0000000		rs2		rs1		111		rd		0110011		AND	rd = rs1 & rs2

I-type

31	20	19	15	14	12	11	7	6	0		
imm[11:0]		rs1		funct3		rd		opcode		Mnemonic	Description
imm[11:0]		rs1		010		rd		0000011		LW	$rd = M[rs1 + imm]$
imm[11:0]		rs1		000		rd		0010011		ADDI	$rd = rs1 + imm$
imm[11:0]		rs1		010		rd		0010011		SLTI	$rd = (rs1_s < imm_s)? 1:0$
imm[11:0]		rs1		011		rd		0010011		SLTIU	$rd = (rs1_u < imm_u)? 1:0$
imm[11:0]		rs1		100		rd		0010011		XORI	$rd = rs1 \wedge imm$
imm[11:0]		rs1		110		rd		0010011		ORI	$rd = rs1 \vee imm$
imm[11:0]		rs1		111		rd		0010011		ANDI	$rd = rs1 \wedge imm$
imm[11:0]		rs1		000		rd		0000011		LB	$rd = M[rs1 + imm]_{bs}$
0000000	shamt	rs1		001		rd		0010011		SLLI	$rd = rs1_u \ll shamt$
0000000	shamt	rs1		101		rd		0010011		SRLI	$rd = rs1_u \gg shamt$
0100000	shamt	rs1		101		rd		0010011		SRAI	$rd = rs1_s \gg shamt$
imm[11:0]		rs1		000		rd		1100111		JALR	$rd = PC + 4$ $PC = imm + rs1$ (Set LSB of PC to 0)
imm[11:0]		rs1		001		rd		0000011		LH	$rd = M[rs1 + imm]_{hs}$
imm[11:0]		rs1		100		rd		0000011		LHU	$rd = M[rs1 + imm]_{hu}$
imm[11:0]		rs1		101		rd		0000011		LBU	$rd = M[rs1 + imm]_{bu}$

Problem 1 – Instructions (2/2)

□ S-type

31	25	24	20	19	15	14	12	11	7	6	0		
imm[11:5]		rs2		rs1		funct3		imm[4:0]		opcode		Mnemonic	Description
imm[11:5]		rs2		rs1		010		imm[4:0]		0100011		SW	$M[rs1+imm] = rs2$
imm[11:5]		rs2		rs1		000		imm[4:0]		0100011		SB	$M[rs1+imm]_b = rs2_b$
imm[11:5]		rs2		rs1		001		imm[4:0]		0100011		SH	$M[rs1+imm]_h = rs2_h$

□ B-type

31	25	24	20	19	15	14	12	11	7	6	0		
imm[12 10:5]		rs2		rs1		funct3		imm[4:1 11]		opcode		Mnemonic	Description
imm[12 10:5]		rs2		rs1		000		imm[4:1 11]		1100011		BEQ	PC = (rs1 == rs2)? PC + imm: PC + 4
imm[12 10:5]		rs2		rs1		001		imm[4:1 11]		1100011		BNE	PC = (rs1 != rs2)? PC + imm: PC + 4
imm[12 10:5]		rs2		rs1		100		imm[4:1 11]		1100011		BLT	PC = (rs1 _s < rs2 _s)? PC + imm: PC + 4
imm[12 10:5]		rs2		rs1		101		imm[4:1 11]		1100011		BGE	PC = (rs1 _s ≥ rs2 _s)? PC + imm: PC + 4
imm[12 10:5]		rs2		rs1		110		imm[4:1 11]		1100011		BLTU	PC = (rs1 _u < rs2 _u)? PC + imm: PC + 4
imm[12 10:5]		rs2		rs1		111		imm[4:1 11]		1100011		BGEU	PC = (rs1 _u ≥ rs2 _u)? PC + imm: PC + 4

□ U-type

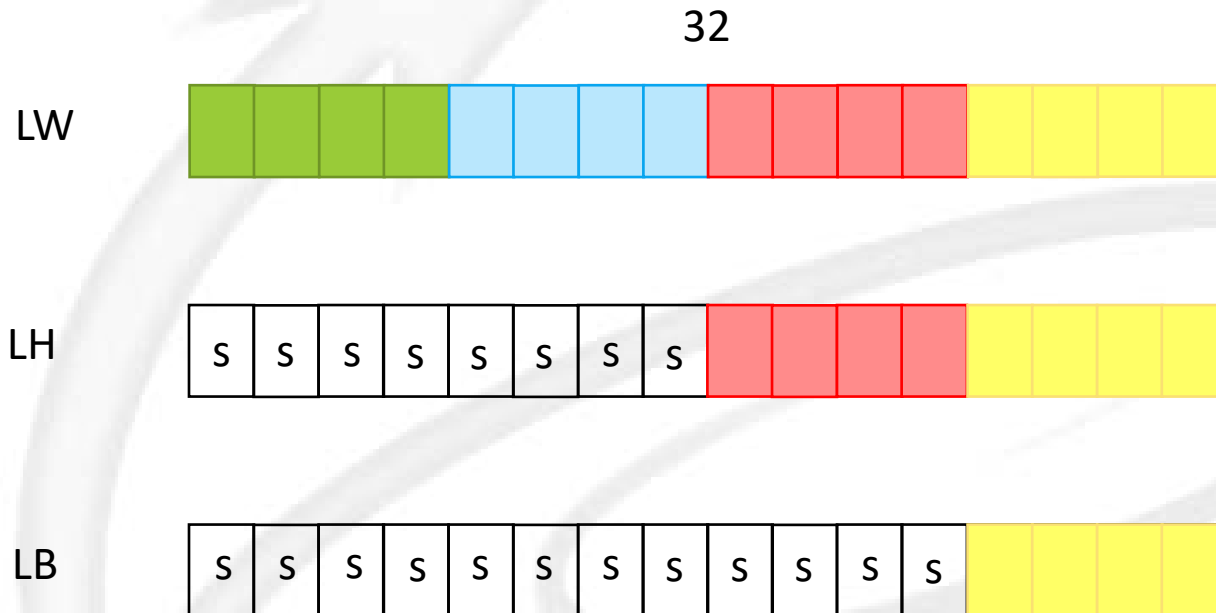
31	12	11	7	6	0		
imm[31:12]		rd		opcode		Mnemonic	Description
imm[31:12]		rd		0010111		AUIPC	rd = PC + imm
imm[31:12]		rd		0110111		LUI	rd = imm

□ J-type

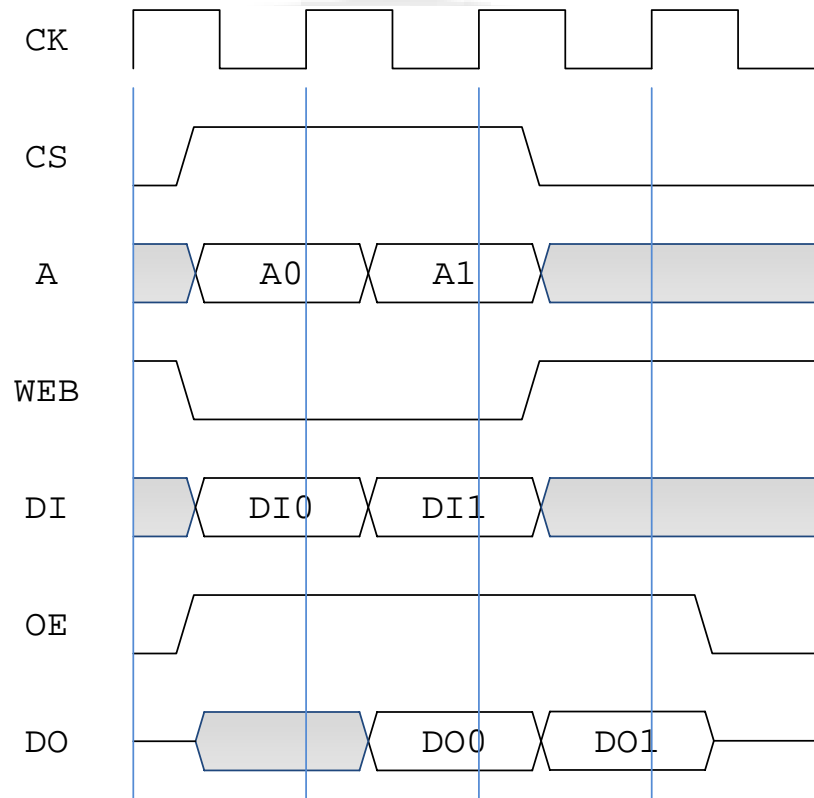
31	12	11	7	6	0		
imm[20 10:1 11 19:12]		rd		opcode		Mnemonic	Description
imm[20 10:1 11 19:12]		rd		1101111		JAL	rd = PC + 4 PC = PC + imm

Problem 1 – Read/Write Word/Half-word/Byte

- If the instruction is used to read half-word or bytes, all other bytes left must be filled with 0s or sign bits.



Problem 1 – SRAM



NOTE: DO delays **1 clock** after Address is imported, be aware of the pipeline registers delay



作業驗證說明

Program

- prog0
 - ➔ 測試37個instruction (助教提供)
- prog1
 - ➔ Sort Algorithm
- prog2
 - ➔ Multiplication
- prog3
 - ➔ Greatest common divisor

Simulation

Table B-1: Simulation commands

Simulation Level	Command
Problem1	
RTL	<code>make rtl_all</code>
Post-synthesis	<code>make syn_all</code>

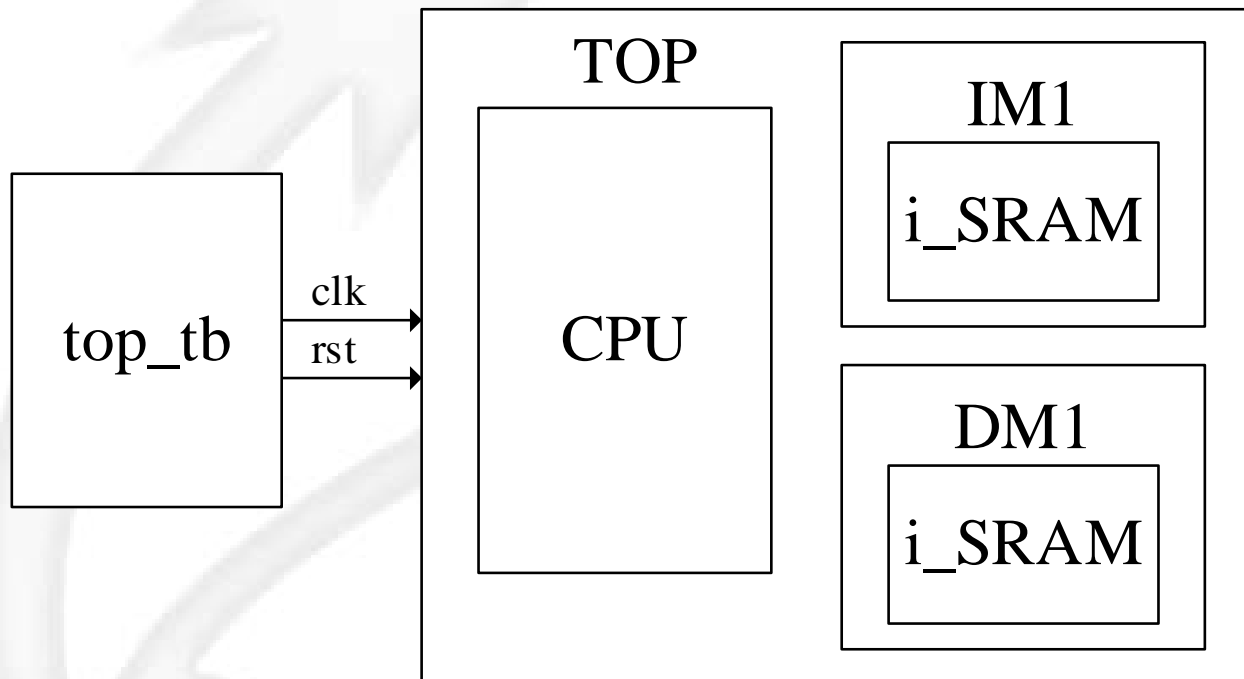
Table B-2: Makefile macros

Situation	Command	Example
RTL simulation for progX	<code>make rtlX</code>	<code>make rtl0</code>
Post-synthesis simulation for progX	<code>make synX</code>	<code>make syn1</code>
Dump waveform (no array)	<code>make {rtlX,synX} FSDB=1</code>	<code>make rtl2 FSDB=1</code>
Dump waveform (with array)	<code>make {rtlX,synX} FSDB=2</code>	<code>make syn3 FSDB=2</code>
Open nWave without file pollution	<code>make nWave</code>	
Open Superlint without file pollution	<code>make superlint</code>	
Open DesignVision without file pollution	<code>make dv</code>	
Synthesize your RTL code (You need write <i>synthesis.tcl</i> in <i>script</i> folder by yourself)	<code>make synthesize</code>	
Delete built files for simulation, synthesis or verification	<code>make clean</code>	
Check correctness of your file structure	<code>make check</code>	
Compress your homework to <i>tar</i> format	<code>make tar</code>	



作業繳交注意事項

Testbench Structure



Module (1/2)

- Module name 須符合下表要求

Category	Name			
	File	Module	Instance	SDF
RTL	top.sv	top	TOP	
RTL	SRAM_wrapper.sv	SRAM_wrapper	IM1	
RTL	SRAM_wrapper.sv	SRAM_wrapper	DM1	
RTL	SRAM_rtl.sv	SRAM	i_SRAM	

- 紫色部分為助教已提供或已定義好，請勿任意更改
- 其餘部分需按照要求命名，以免testbench抓不到正確的名稱

Module (2/2)

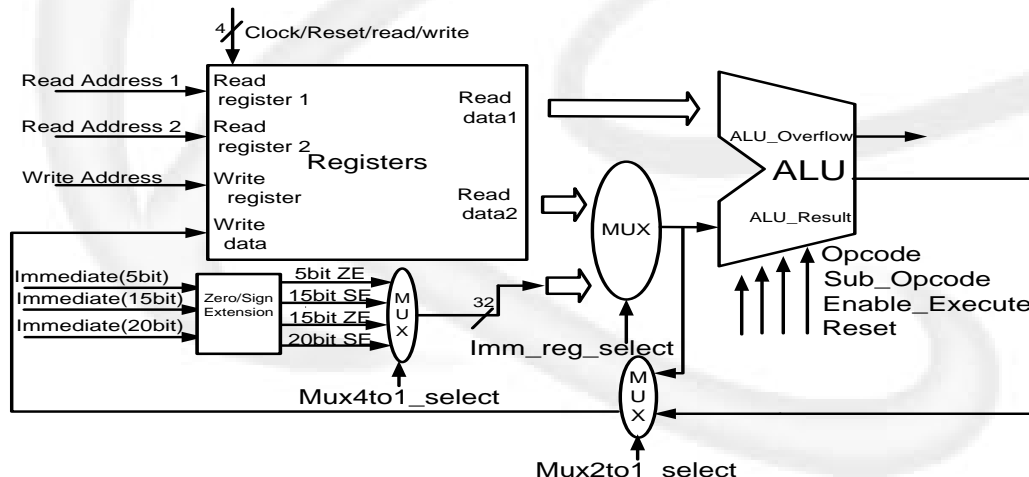
Module port 須符合下表要求

Module	Specifications			
top	Name	Signal	Bits	Function explanation
	clk	input	1	System clock
	rst	input	1	System reset (active high)
SRAM_wrapper	CK	input	1	System clock
	CS	input	1	Chip select (active high)
	OE	input	1	Output enable (active high)
	WEB	input	4	Write enable (active low)
	A	input	14	Address
	DI	input	32	Data input
	DO	output	32	Data output
SRAM	Memory Space			
	Memory_byte0	logic	8	Size: [16384]
	Memory_byte1	logic	8	Size: [16384]
	Memory_byte2	logic	8	Size: [16384]
	Memory_byte3	logic	8	Size: [16384]

- 紫色部分為助教已提供或已定義好，請勿任意更改
- 其餘部分需按照要求命名，以免testbench抓不到正確的名稱

Report (1/2)

- ❑ 請勿將code貼在.docx內
 - ➔ 請將.sv包在壓縮檔內，不可截圖於.docx中
- ❑ 需要Summary及Lessons learned
- ❑ Block diagram
 - ➔ 不必畫到gate level，除非該處的邏輯對於設計上有重要意義
 - ➔ 可用一個矩形標上名稱以及I/O代表一個 functional block
 - ➔ 呈現要點在於讓人較容易理解你的設計的架構
 - ➔ 可以使用Visio、Open Office Draw，或其他繪圖軟體



Report (2/2)

□ 驗證波形圖

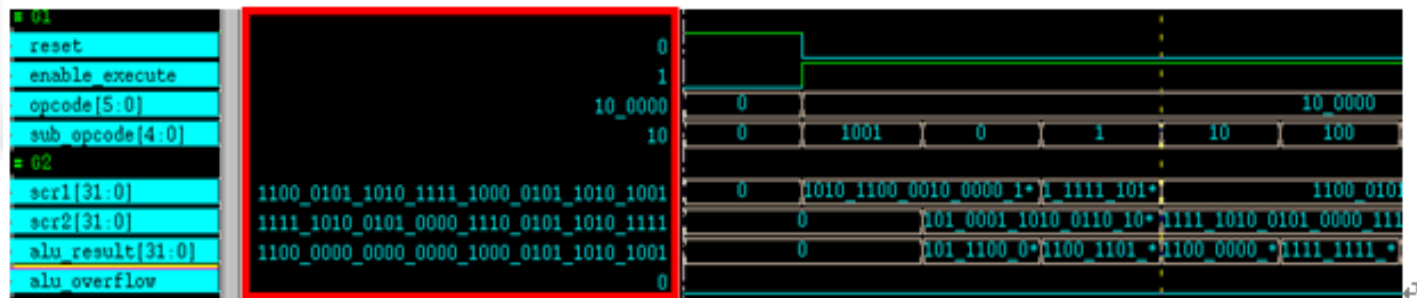
- 保留完整訊號名稱以及訊號值
- 輔以文字解釋該波形圖的操作
- 可在波形圖加上標示輔助了解
- 截圖裁減至合適大小

v. 測試AND的功能。當enable_execute為1時，表示alu開始做運算，opcode為100000，sub_opcode為00010，動作為AND。

scr1=32' b1100_0101_1010_1111_1000_0101_1010_1001，

scr2=32' b1111_1010_0101_0000_1110_0101_1010_1111，

結果為=32' b1100_0000_0000_0000_1000_0101_1010_1001。



繳交檔案

- 依照檔案結構壓縮成 “.tar” 格式
 - ➔ 在Homework主資料夾(N260XXXXX)使用make tar產生的tar檔即可符合要求
- 檔案結構請依照作業說明
- 請勿附上檔案結構內未要求繳交的檔案
 - ➔ 在Homework主資料夾(N260XXXXX)使用make clean即可刪除不必要的檔案
- 請務必確認繳交檔案可以在SoC實驗室的工作站下compile，且功能正常
- 無法compile將直接以0分計算
- 請勿使用generator產生code再修改
- 禁止抄襲

檔案結構 (1/2)

- N260XXXXX.docx
→ Your report file
- src
→ Your source code (*.sv)
- include
→ Your definition code (*.svh)
- StudentID
→ Specify your Student ID number
- sim/CYCLE
→ Specify your clock cycle time
- sim/MAX
→ Specify max clock cycle number

Fig. A-1 File hierarchy

- N260XXXXX.tar (Don't add version text in filename, e.g. N260XXXXX_v1.tar)
- N260XXXXX (Main folder of this homework)
 - N260XXXXX.docx (Your homework report)
 - StudentID (Specify your student ID number in this file)
 - Makefile (You shouldn't modify it)
 - src (Your RTL code with sv format)
 - top.sv
 - SRAM_wrapper.sv
 - Other submodules (*.sv)
 - include (Your RTL definition with svh format, optional)
 - Definition files (*.svh)
 - syn (Your synthesized code and timing file, optional)
 - top_syn.v
 - top_syn.sdf
 - script (Any scripts of verification, synthesis or place and route)
 - Script files (*.sdc, *.tcl or *.setup)
 - sim (Testbenches and memory libraries)
 - top_tb.sv (Main testbench. You shouldn't modify it)
 - CYCLE (Specify your clock cycle time in this file)
 - MAX (Specify max clock cycle number in this file)
 - SRAM (SRAM libraries and behavior models)
 - Library files (*.lib, *.db, *.lef or *.gds)
 - SRAM.ds (SRAM datasheet)
 - SRAM_rtl.sv (SRAM RTL model)
 - SRAM.v (SRAM behavior model)

檔案結構 (2/2)

- sim/prog0
 - ➔ Don't modify contents
- sim/progX ($X \neq 0$)
 - ➔ main.S
 - ➔ main.c
 - ◆ Submit one of these

- 📁 **SRAM** (SRAM libraries and behavior models)
 - 📄 Library files (*.lib, *.db, *.lef or *.gds)
 - 📄 **SRAM.ds** (SRAM datasheet)
 - 📄 **SRAM_rtl.sv** (SRAM RTL model)
 - 📄 **SRAM.v** (SRAM behavior model)
- 📁 **prog0** (Subfolder for Program 0)
 - 📄 **Makefile** (Compile and generate memory content)
 - 📄 **main.S** (Assembly code for verification)
 - 📄 **setup.S** (Assembly code for testing environment setup)
 - 📄 **link.ld** (Linker script for testing environment)
 - 📄 **golden.hex** (Golden hexadecimal data)
- 📁 **prog1** (Subfolder for Program 1)
 - 📄 **Makefile** (Compile and generate memory content)
 - 📄 **main.S** * (Assembly code for verification)
 - 📄 **main.c** * (C code for verification)
 - 📄 **data.S** (Assembly code for testing data)
 - 📄 **setup.S** (Assembly code for testing environment setup)
 - 📄 **link.ld** (Linker script for testing environment)
 - 📄 **golden.hex** (Golden hexadecimal data)
- 📁 **prog2** (Subfolder for Program 2)
 - 📄 **Makefile** (Compile and generate memory content)
 - 📄 **main.S** * (Assembly code for verification)
 - 📄 **main.c** * (C code for verification)
 - 📄 **data.S** (Assembly code for testing data)
 - 📄 **setup.S** (Assembly code for testing environment setup)
 - 📄 **link.ld** (Linker script for testing environment)
 - 📄 **golden.hex** (Golden hexadecimal data)
- 📁 **prog3** (Subfolder for Program 3)
 - 📄 **Makefile** (Compile and generate memory content)
 - 📄 **main.S** * (Assembly code for verification)
 - 📄 **main.c** * (C code for verification)
 - 📄 **data.S** (Assembly code for testing data)
 - 📄 **setup.S** (Assembly code for testing environment setup)
 - 📄 **link.ld** (Linker script for testing environment)
 - 📄 **golden.hex** (Golden hexadecimal data)

繳交期限

- 2021/10/20 (三) 14:00前上傳
 - ➔ 不接受遲交，請務必注意時間
 - ➔ Moodle只會留存你最後一次上傳的檔案，檔名只要是「**N260XXXXX.tar**」即可，不需要加上版本號
- 作業二預計在2021/10/13 (三)會上傳，請務必加快作業一的設計時間，以免壓縮到作業二的時間

注意事項

- 本次作業合成的部分有計入PA(Performance & Area)，表現較為優異者將有較高的評分(5%)
- 作業部分有任何問題請在moodle上的作業討論區發問並可參考其他人是否有類似的問題，助教信箱恕不回覆。
- 在Script/DC.sdc中，可以更改clk period範圍，請注意最大的接受值為20.0，超過此範圍者恕不納入計分。此外，若有更改預設clock period者，請務必更改set_input_delay -max至1/2 clk period，以利後續作業的合成及模擬。
- 本次作業需在SoC環境下模擬，請同學務必在SoC教室執行模擬，若是助教在評分作業時於SoC無法成功模擬則以0分計算。

**Thanks for your participation and
attendance !!**