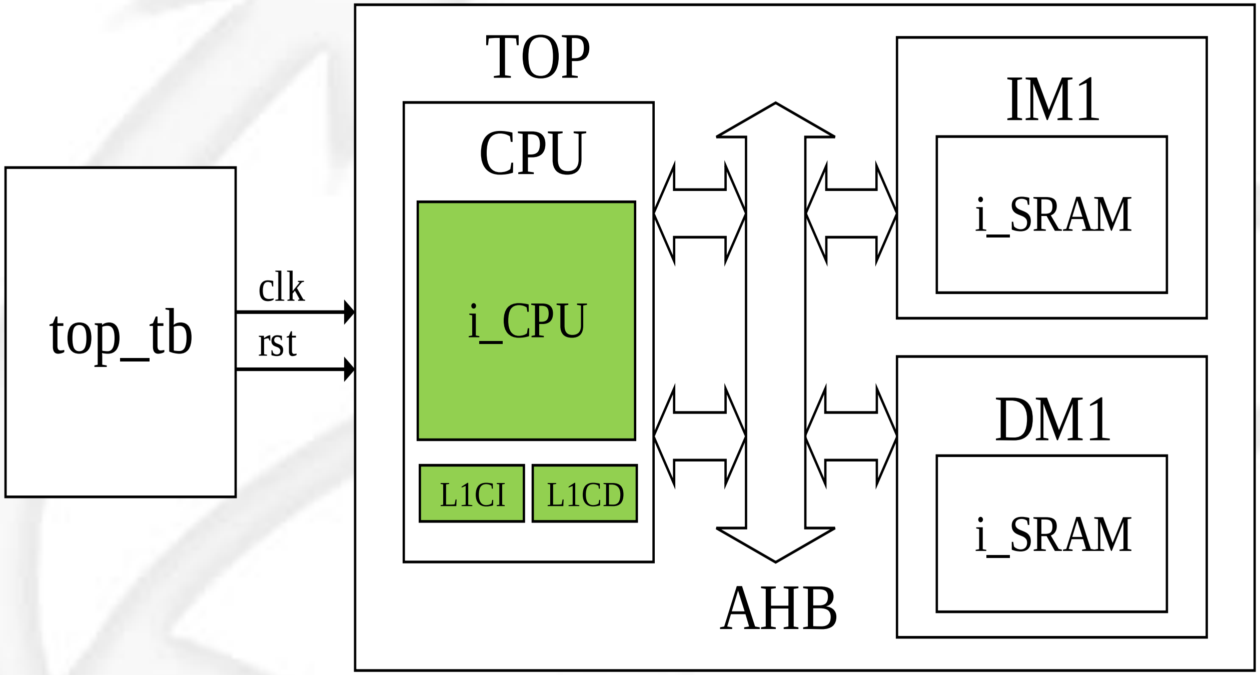
Homework Ⅲ 說明

Instructor：Lih-Yih Chiou

TA : Claire

Date : 2018/11/01

System structure



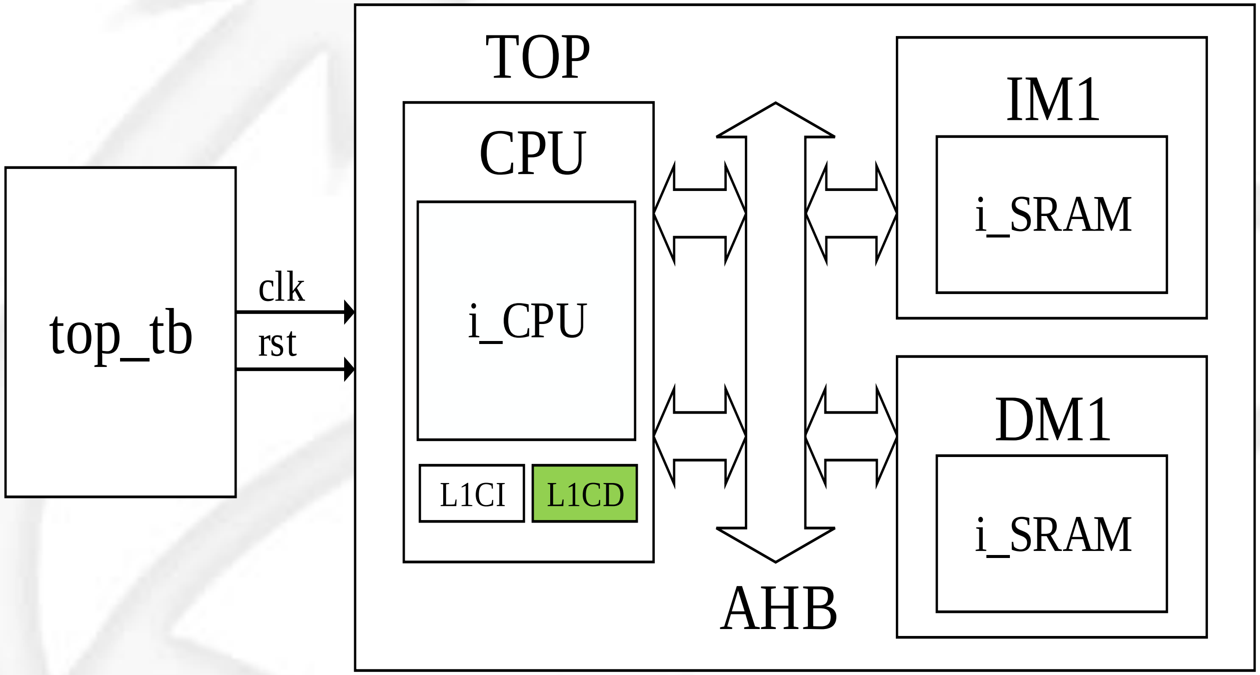
Outline

- Problem 1
  - Specification
  - Verification
- Problem 2
  - Specification
  - Verification
- Submission rule

Problem 1

Specification

Specification(1/4)



Specification(2/4)

L1C_data	clk	input	1	clock
	rst	input	1	reset (active high)
	core_addr	input	32	address from CPU
	core_req	input	1	memory access request from CPU
	core_write	input	1	write signal from CPU
	core_in	input	32	data from CPU
	core_type	input	3	write/read byte, half word, or word (listed in include/def.svh) from CPU
	D_out	input	32	data from CPU wrapper
	D_wait	input	1	wait signal from CPU wrapper
	core_out	output	32	data to CPU
	core_wait	output	1	wait signal to CPU
	D_req	output	1	request to CPU wrapper
	D_addr	output	32	address to CPU wrapper
	D_write	output	1	write signal to CPU wrapper
	D_in	output	32	write data to CPU wrapper
	D_type	output	3	write/read byte, half word, or word (listed in include/def.svh)
	valid	logic	64	valid bits of each tag
	hit	logic	1	equal 1 when valid && TA_out equal core_addr[31:10]; otherwise, equal 0
	index	logic	6	address to array
	TA_write	logic	1	write signal to tag_array
	TA_read	logic	1	read signal to tag_array
	TA_in	logic	22	write data to tag_array
	TA_out	logic	22	read data from tag_array
	DA_write	logic	16	write data to data_array
	DA_read	logic	1	read signal to data_array
	DA_in	logic	128	write data to data_array
	DA_out	logic	128	read data from data_array

Inputs and outputs of module L1C_data has declared in src/L1C_data.sv. You can only add logics inside the modules.

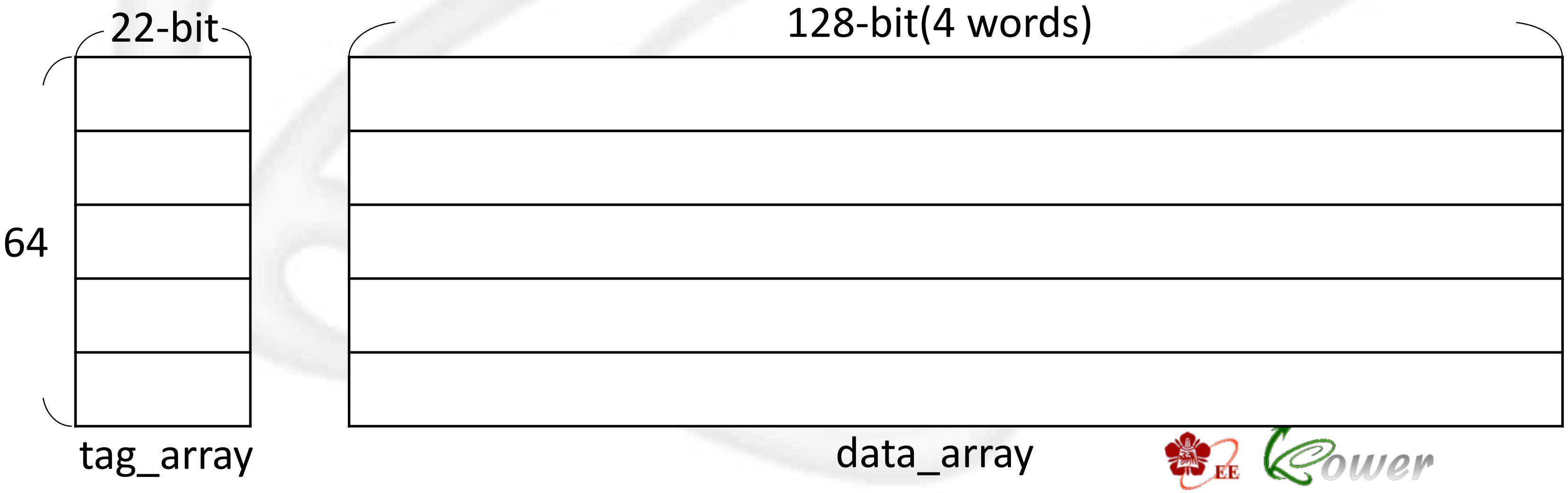
Specification(3/4)

Table 1-1: Data cache actions on different condition

condition	core read	core write
hit	transmit data into core	write data into D-cache and DM
miss	read a line from DM	only write data into DM

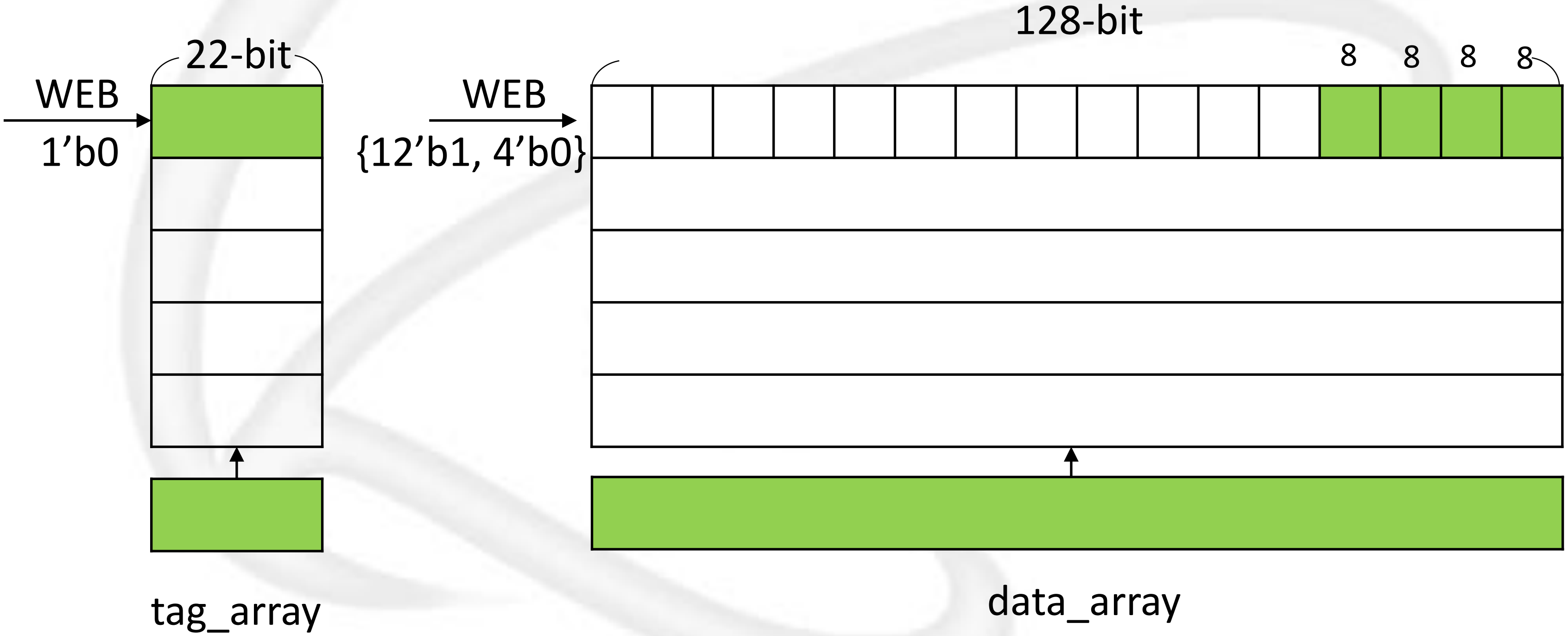
Table 1-2: Array characteristics

Type	Word	Bits per byte	Bytes	Bits per line	Writing mode
data_array	64	8	16	128	Byte Write
tag_array		22	N/A	22	Word Write



Specification(4/4)

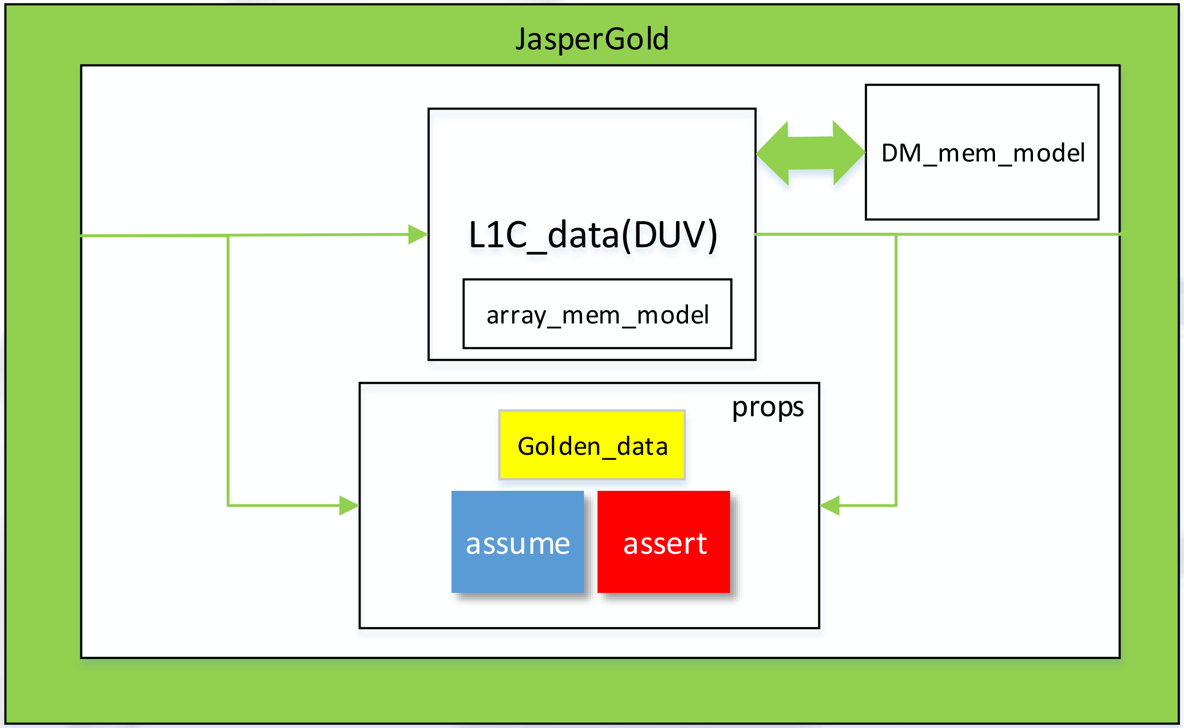
- Tag_array is word write, while data_array is byte write
- Refer to “sim/tag_array/tag_array.ds” and “sim/data_array/data_array.ds” for details



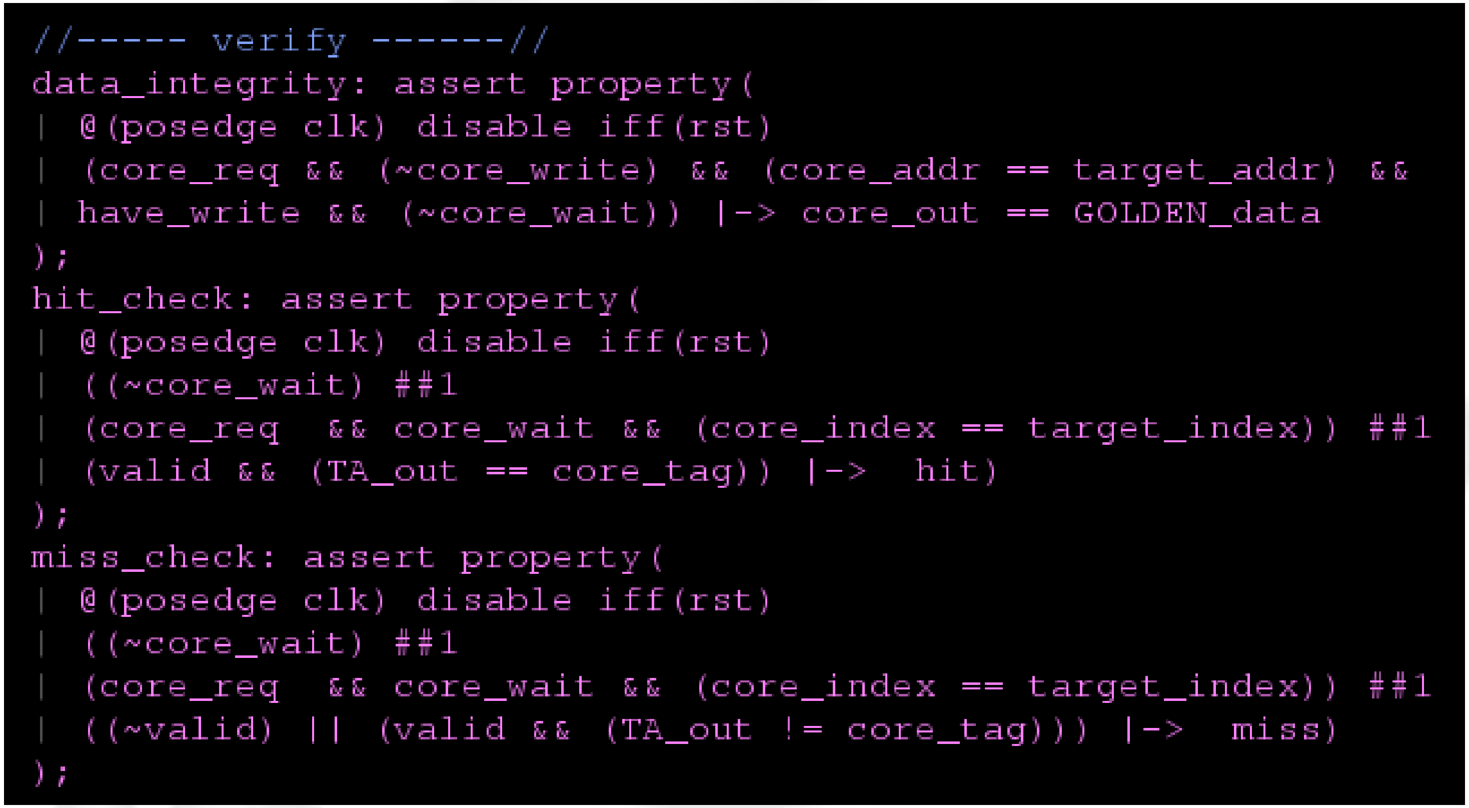
Problem 1

Verification

Verification(1/3)



Verification(2/3)



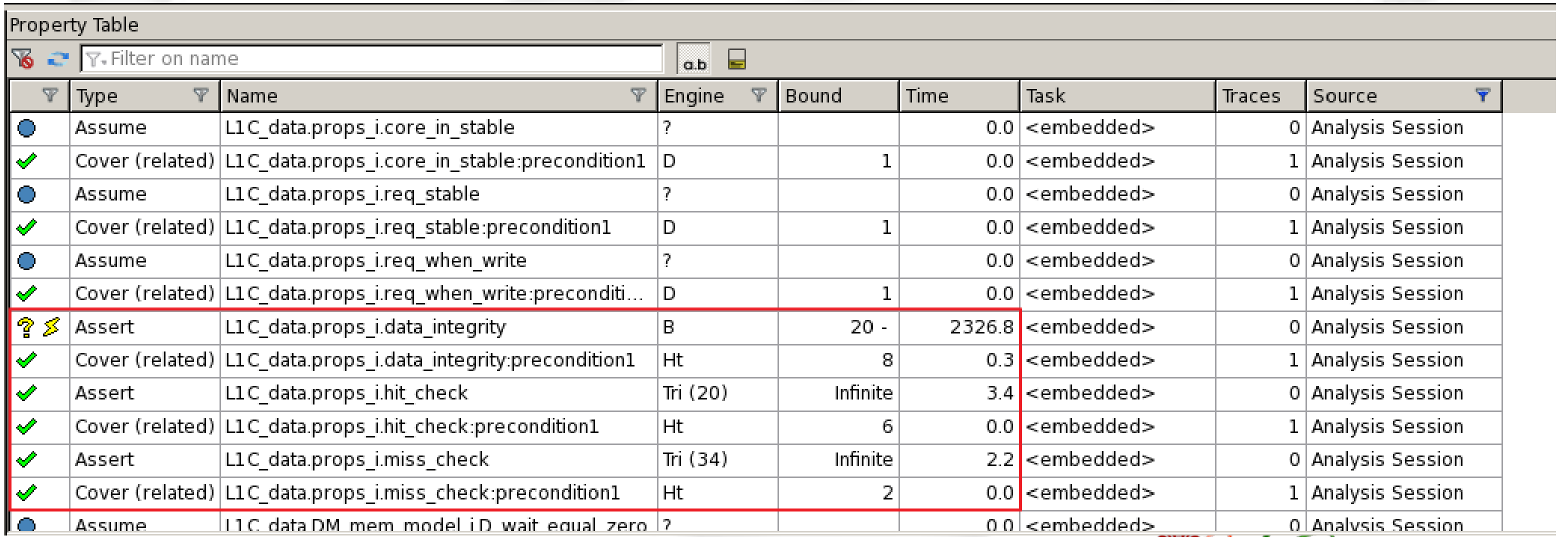
```
//----- verify ------//
data_integrity: assert property(
|  @(posedge clk) disable iff(rst)
|  (core_req && (~core_write) && (core_addr == target_addr) &&
|  have_write && (~core_wait)) |-> core_out == GOLDEN_data
);
hit_check: assert property(
|  @(posedge clk) disable iff(rst)
|  ((~core_wait) ##1
|  (core_req  && core_wait && (core_index == target_index)) ##1
|  (valid && (TA_out == core_tag)) |->  hit)
);
miss_check: assert property(
|  @(posedge clk) disable iff(rst)
|  ((~core_wait) ##1
|  (core_req  && core_wait && (core_index == target_index)) ##1
|  ((~valid) || (valid && (TA_out != core_tag))) |->  miss)
);
```

Bonus : other cache assertions which increase code coverage

Verification(3/3)

- Don’t modify file under sva/ directory, except for new assertions for bonus
- Use JasperGold to verify L1 data cache by command “make jg”
- Show three assertions (and bonus)with ✓ ,

  or ⚡ but time over 1800 seconds

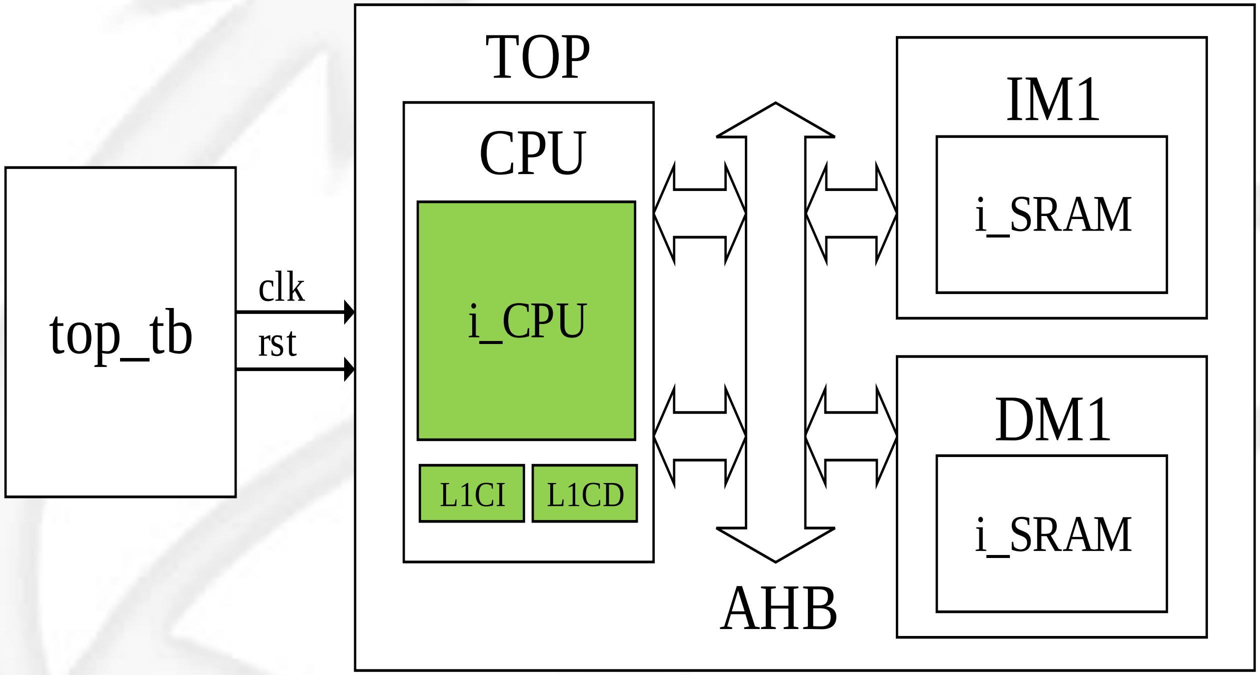


	Type	Name	Engine	Bound	Time	Task	Traces	Source
●	Assume	L1C_data.props_i.core_in_stable	?		0.0	<embedded>	0	Analysis Session
✓	Cover (related)	L1C_data.props_i.core_in_stable:precondition1	D	1	0.0	<embedded>	1	Analysis Session
●	Assume	L1C_data.props_i.req_stable	?		0.0	<embedded>	0	Analysis Session
✓	Cover (related)	L1C_data.props_i.req_stable:precondition1	D	1	0.0	<embedded>	1	Analysis Session
●	Assume	L1C_data.props_i.req_when_write	?		0.0	<embedded>	0	Analysis Session
✓	Cover (related)	L1C_data.props_i.req_when_write:preconditi…	D	1	0.0	<embedded>	1	Analysis Session
? ⚡	Assert	L1C_data.props_i.data_integrity	B	20 -	2326.8	<embedded>	0	Analysis Session
✓	Cover (related)	L1C_data.props_i.data_integrity:precondition1	Ht	8	0.3	<embedded>	1	Analysis Session
✓	Assert	L1C_data.props_i.hit_check	Tri (20)	Infinite	3.4	<embedded>	0	Analysis Session
✓	Cover (related)	L1C_data.props_i.hit_check:precondition1	Ht	6	0.0	<embedded>	1	Analysis Session
✓	Assert	L1C_data.props_i.miss_check	Tri (34)	Infinite	2.2	<embedded>	0	Analysis Session
✓	Cover (related)	L1C_data.props_i.miss_check:precondition1	Ht	2	0.0	<embedded>	1	Analysis Session
●	Assume	L1C_data.DM_mem_model_i.D_wait_equal_zero	?		0.0	<embedded>	0	Analysis Session

Problem 2

Specification

Specification(1/5)



Specification(2/5)

- Implement 6 new instructions on your CPU

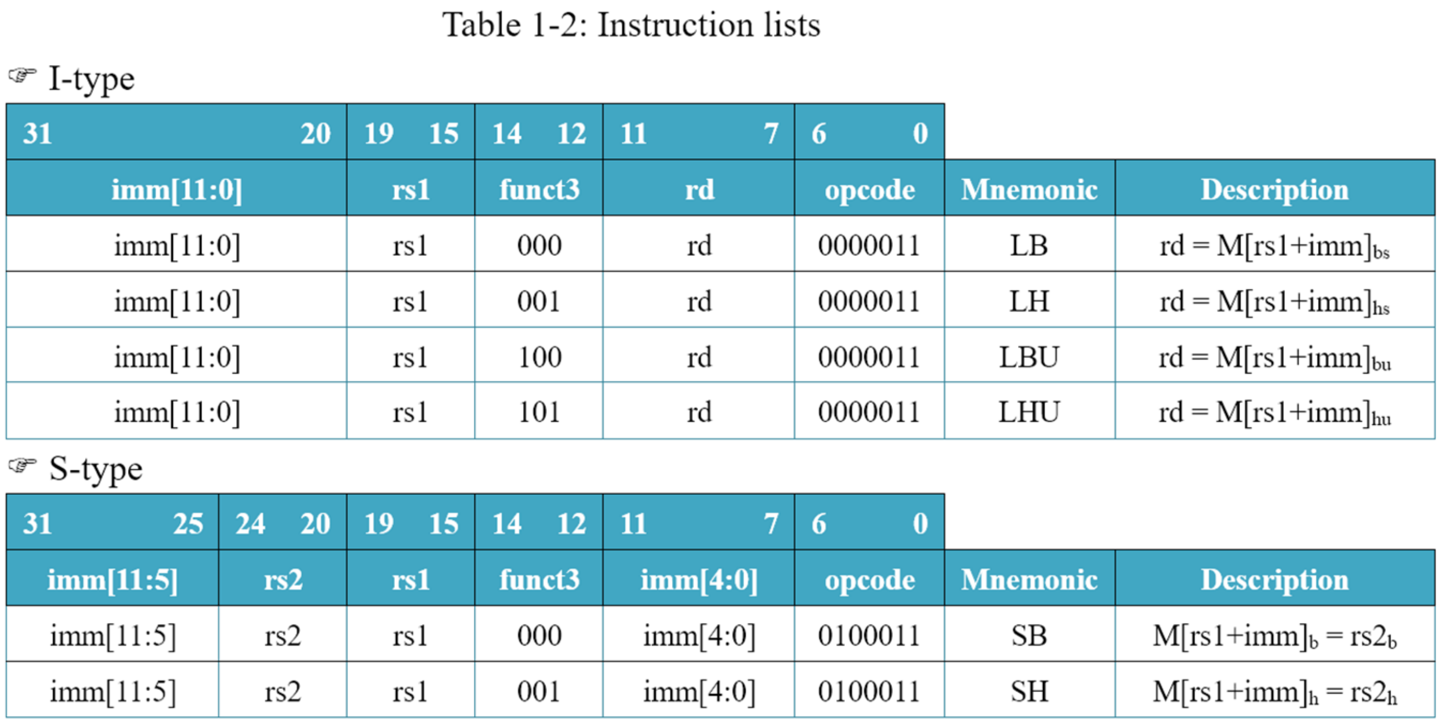


Table 1-2: Instruction lists

☞ 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	000	rd	0000011	LB	$rd = M[rs1+imm]_{bs}$
imm[11:0]	rs1	001	rd	0000011	LH	$rd = M[rs1+imm]_{hs}$
imm[11:0]	rs1	100	rd	0000011	LBU	$rd = M[rs1+imm]_{bu}$
imm[11:0]	rs1	101	rd	0000011	LHU	$rd = M[rs1+imm]_{hu}$

☞ 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	000	imm[4:0]	0100011	SB	$M[rs1+imm]_{b} = rs2_{b}$
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH	$M[rs1+imm]_{h} = rs2_{h}$

1. Byte = 8 bits
2. half-word = 16 bits
3. SB and SH means store lowest byte or half-word of rs2 to the memory

Specification(3/5)

- Module name須符合下表要求

Category	Name			
	File	Module	Instance	SDF
RTL	top.sv	top	TOP	
Gate-Level	top_syn.v	top	TOP	top_syn.sdf
Physical	top_pr.sv	top	TOP	top_pr.sdf
RTL	L1C_inst.sv	L1C_inst	L1CI	
RTL	L1C_data.sv	L1C_data	L1CD	
RTL	AHB.sv	AHB	AHB1	
RTL	SRAM_wrapper.sv	SRAM_wrapper	IM1	
RTL	SRAM_wrapper.sv	SRAM_wrapper	DM1	
RTL	SRAM_rtl.sv	SRAM	i_SRAM	
RTL	tag_array_wrapper.sv	tag_array_wrapper	TA	
RTL	tag_array_rtl.sv	tag_array	i_tag_array	
RTL	data_array_wrapper.sv	data_array_wrapper	DA	
RTL	data_array_rtl.sv	data_array	i_data_array	

- 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

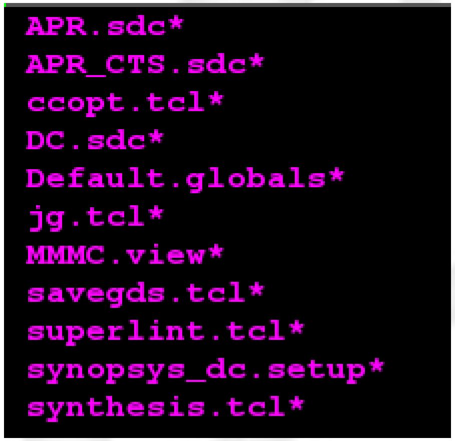
Specification(4/5)

- Module port須符合下表要求

Module	Specifications			
	Name	Signal	Bits	Function explanation
top	clk	input	1	System clock
	rst	input	1	System reset (active high)
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]
L1C_inst /L1C_data	clk	input	1	clock
	rst	input	1	reset (active high)
	core_addr	input	32	address from CPU
	core_req	input	1	memory access request from CPU
	core_write	input	1	write signal from CPU
	core_in	input	32	data from CPU
	core_type	input	3	write/read byte, half word , or word (listed in include/def.svh) from CPU
	I_out/D_out	input	32	data from CPU wrapper
	I_wait/D_wait	input	1	wait signal from CPU wrapper
	core_out	output	32	data to CPU
	core_wait	output	1	wait signal to CPU
	I_req/D_req	output	1	request to CPU wrapper
	I_addr/D_addr	output	32	address to CPU wrapper
	I_write/ D_write	output	1	write signal to CPU wrapper
	I_in/D_in	output	32	write data to CPU wrapper
	I_type/D_type	output	3	write/read byte, half word, or word (listed in include/def.svh)
	valid	logic	64	valid bits of each tag
	hit	logic	1	equal 1 when valid && TA_out equal core_addr[31:10]; otherwise, equal 0
	index	logic	6	address to array
	TA_write	logic	1	write signal to tag_array
	TA_read	logic	1	read signal to tag_array
	TA_in	logic	22	write data to tag_array
	TA_out	logic	22	read data from tag_array
	DA_write	logic	16	write data to data_array
	DA_read	logic	1	read signal to data_array
	DA_in	logic	128	write data to data_array
	DA_out	logic	128	read data from data_array

Specification(5/5)

- Tag_array.v and data_array.v already include in top_tb.sv, don’t re-include in your design
- Use files in script/ for synthesis and APR



```
APR.sdc*
APR_CTS.sdc*
ccopt.tcl*
DC.sdc*
Default.globals*
jg.tcl*
MMMC.view*
savegds.tcl*
superlint.tcl*
synopsys_dc.setup*
synthesis.tcl*
```

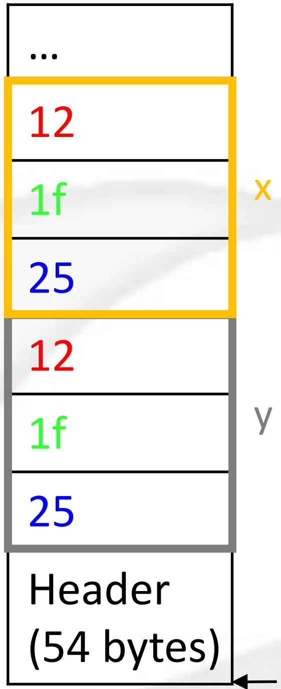
Problem 2

Verification

Verification(1/2)

- prog0
  - 測試31+6個instruction (助教提供)
- prog1
  - Sort algorithm of half-word
- Prog2
  - Gray scale

Verification(2/2)



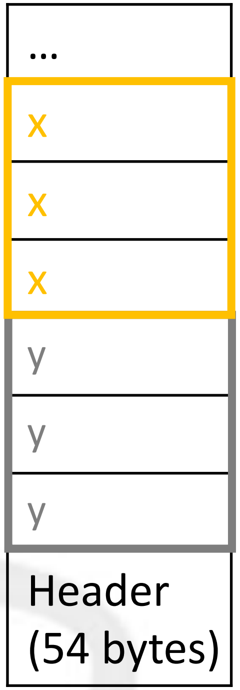
← _binary_image_bmp_start

$x = \mathbf{0.11*b + 0.59*g + 0.3*r}$

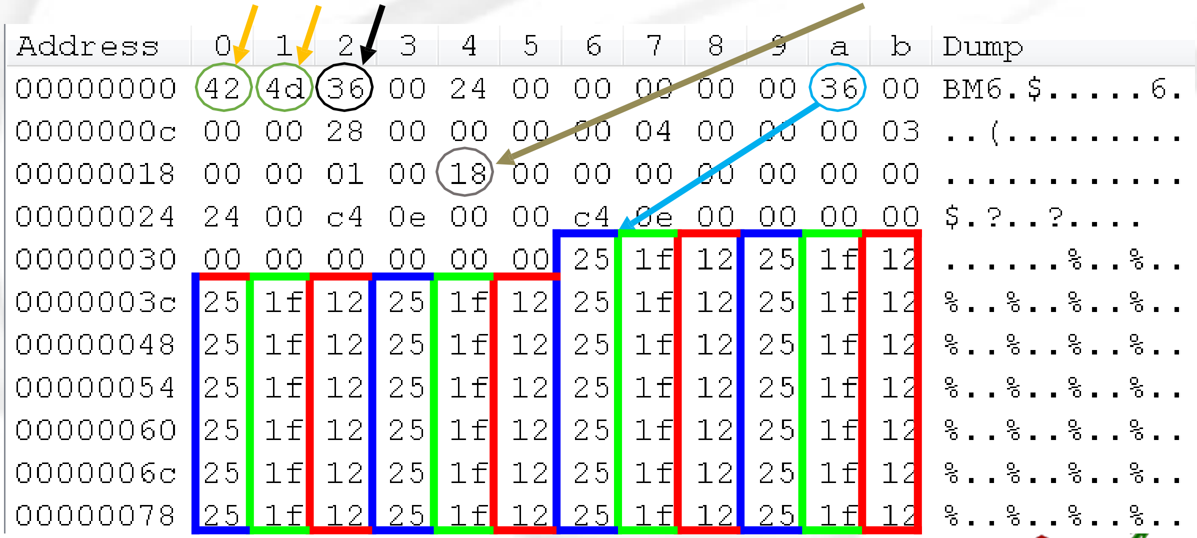
$= 0.11*25 + 0.59*1f + 0.3*\ 12$

$y = \mathbf{0.11*b + 0.59*g + 0.3*r}$

$= 0.11*25 + 0.59*1f + 0.3*\ 12$



B M Size of BMP file (byte) The number of bits per pixel



```
Address    0  1  2  3  4  5  6  7  8  9  a  b  Dump
00000000  42 4d 36 00 24 00 00 00 00 00 36 00  BM6.$.....6.
0000000c  00 00 28 00 00 00 00 04 00 00 00 03  ..(.........
00000018  00 00 01 00 18 00 00 00 00 00 00 00  ............
00000024  24 00 c4 0e 00 00 c4 0e 00 00 00 00  $.?..?....
00000030  00 00 00 00 00 00 25 1f 12 25 1f 12  ......%..%..
0000003c  25 1f 12 25 1f 12 25 1f 12 25 1f 12  %..%..%..%..
00000048  25 1f 12 25 1f 12 25 1f 12 25 1f 12  %..%..%..%..
00000054  25 1f 12 25 1f 12 25 1f 12 25 1f 12  %..%..%..%..
00000060  25 1f 12 25 1f 12 25 1f 12 25 1f 12  %..%..%..%..
0000006c  25 1f 12 25 1f 12 25 1f 12 25 1f 12  %..%..%..%..
00000078  25 1f 12 25 1f 12 25 1f 12 25 1f 12  %..%..%..%..
```

Submission rule

Report

- 請使用附在檔案內的Submission Cover
- 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
- 若兩人為一組，須寫出貢獻度
  - Ex: A(N26071234) 55%, B(N26075678) 45%
  - Total 100%
  - 自己一組則不用寫

繳交檔案 (1/2)

- 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

繳交檔案 (2/2)

- 若兩人為一組，只需一個人上傳作業到Moodle
  - 兩人都上傳會斟酌扣分
- 若兩人為一組，壓縮檔、主資料夾名稱、Report名稱、StudentID檔案內的學號都要為上傳者的學號，另一位只需在StudentID2及Submission Cover內寫上自己的學號。
  - Ex: A(N26071234)負責上傳，組員為B(N26075678)
  - N26071234.tar (壓縮檔)
    - N26071234 (主資料夾)
      - N26071234.docx (Report，Cover寫上兩者的學號)
        - StudentID (裡面填上N26071234)
        - StudentID2 (裡面填上N26075678)
- 自己一組請直接刪除StudentID2檔案

檔案結構

- 參考 Appendix A
- sim/CYCLE
  - Specify your clock cycle time
- sim/MAX
  - Specify max clock cycle number
- sim/prog0
  - Don’t modify contents
- sim/progX (X ≠ 0)
  - main.S
  - main.c
    - Submit one of these

繳交期限

- 2018/11/29 (四) 9:00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

Thanks for your participation and attendance !!