



# 邏輯系統實習

## 實驗十：期末專題

期末專題：*Single cycle ARM32 指令集架構實現*

國立成功大學 電機系

2014

# 電腦系統簡介(1/2)

◆ 電腦系統可簡單分為三層：

◆ Applications software :

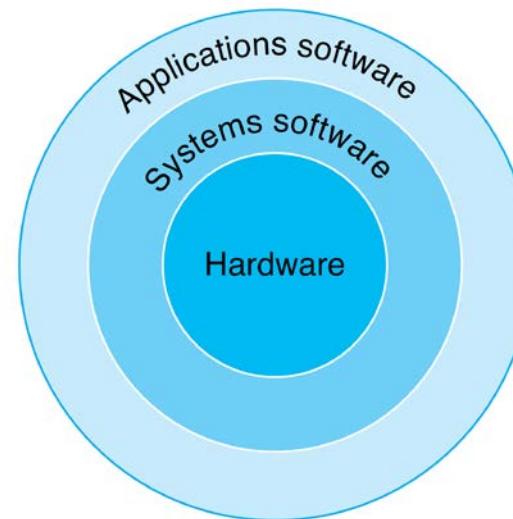
◆ 我們日常使用的應用軟體。Ex : word、Powerpoint...

◆ Systems software :

◆ 管理和溝通硬體的程式。Ex : windows、driver...

◆ Hardware :

◆ 執行指令計算或輸入輸出工具。Ex : CPU、memory...

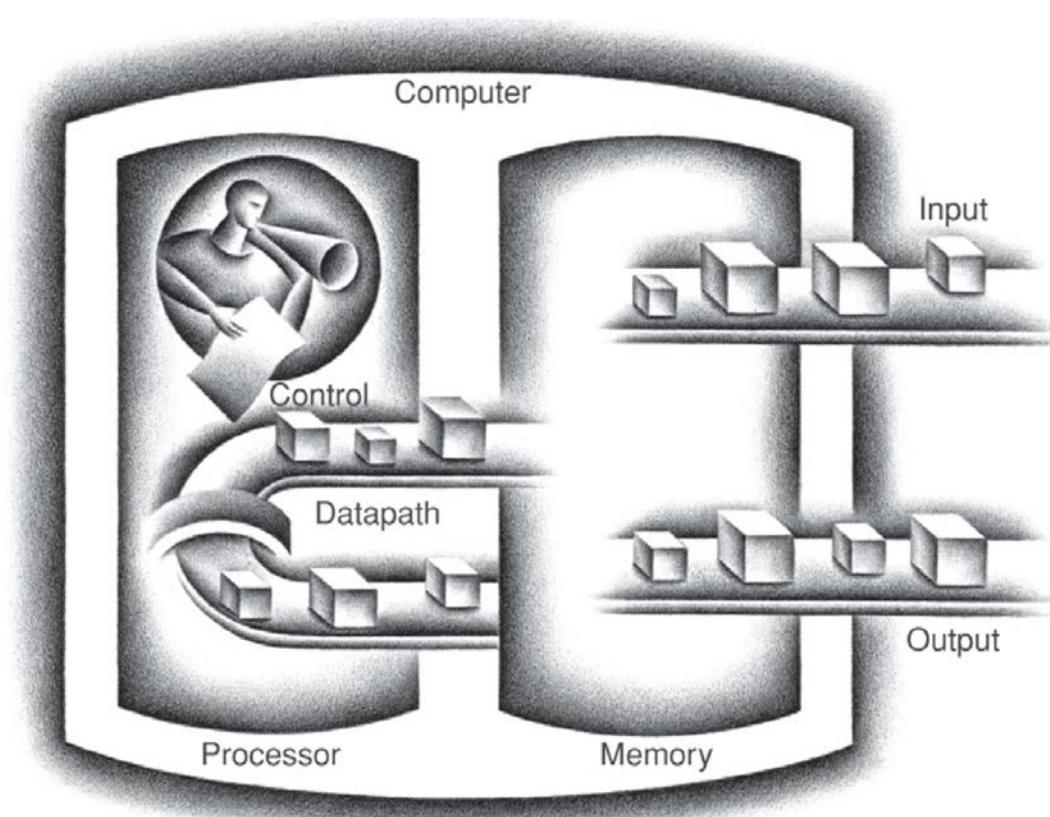


A simplified view of hardware and software as hierarchical layers

# 電腦系統簡介(2/2)

## ◆組成電腦的五大元素：

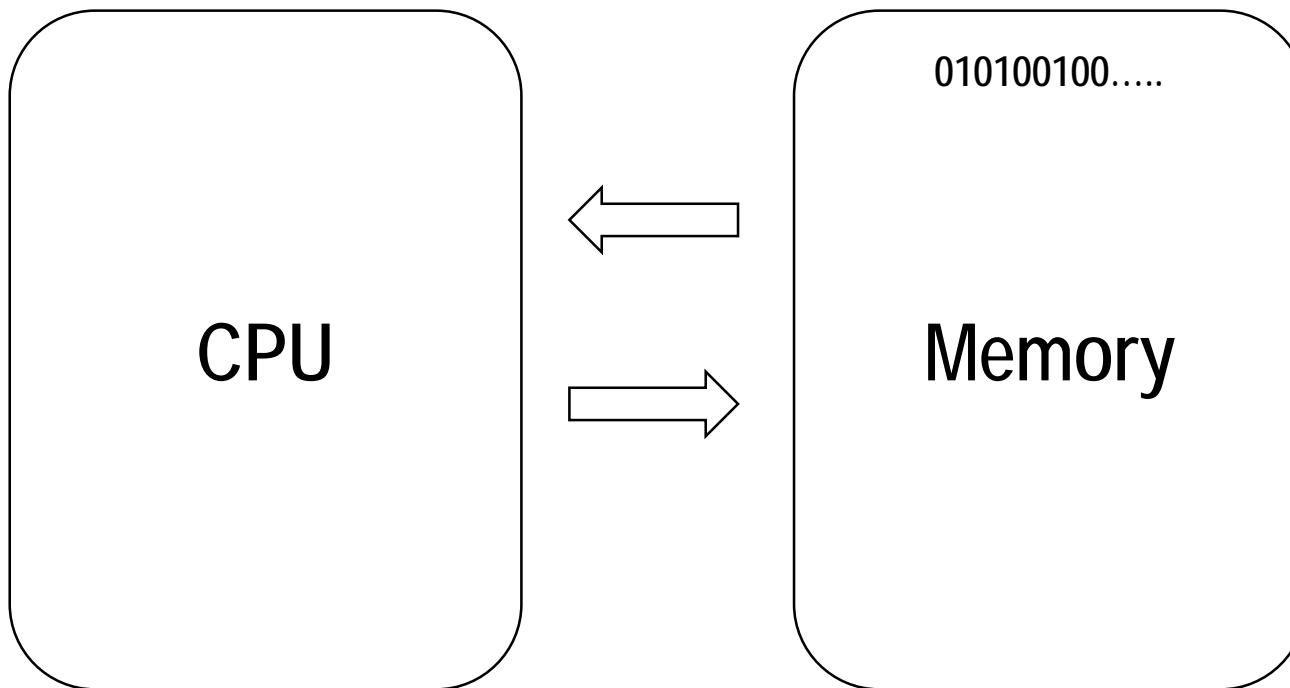
- ◆ 輸入、輸出、記憶體、計算單元、控制單元。
- ◆ 計算單元和控制單元又稱處理器。



The organization of a computer, showing the five classic components.

# 處理器簡介(1/3)

- ◆ CPU永遠就做一件事情：
  - ◆ 抓取指令 → 執行指令 → 抓取指令 → 執行指令.....





# 處理器簡介(2/3)

◆CPU只看得懂0101...訊號，

◆我們常打的高階語言必須轉為二進位的機器碼存入記憶體，CPU才可以抓取指令並且執行。

```
Swap(int v[], int k)
{ int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

compiler

```
swap:
    multi $2, $5,4
    add $2, $4,$2
    lw $15, 0($2)
    lw $16, 4($2)
    sw $16, 0($2)
    sw $15, 4($2)
    jr $31
```

assembler

```
000000001010001000000000100011000
00000000100000100001000001000010
1000110111000100000000000000000000
10001110000100100000000000000000000
10101110000100100000000000000000000
10101101110001000000000000000000000
000000111100000000000000000000000000
```

High-level language  
program (in C)

Assembly language  
program (for MIPS)

Binary machine language  
program (for MIPS)

# 處理器簡介(3/3)

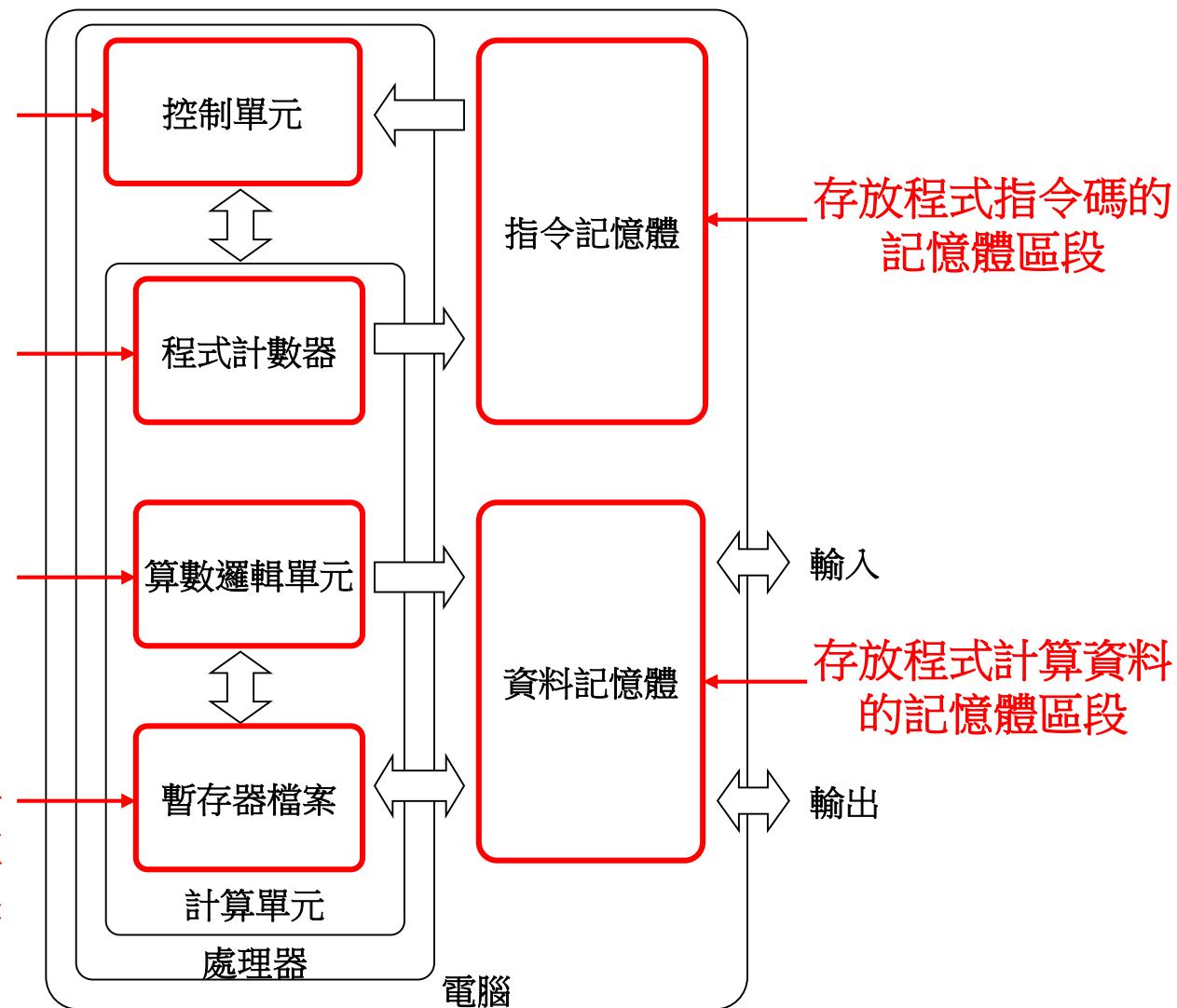
## ◆ 處理器架構

解讀指令並對資料路徑發出控制訊號

指向處理器下一個時脈要執行的指令位址

處理器中的核心計算單元，包含算數與邏輯等運算

處理器中存取資料最快的地方，用來存放暫時計算結果





# ARM微處理器概述(1/2)

## ◆ ARM (Advanced RISC Machines)

- ◆ 是一個公司的名字，也是一類微處理器的通稱。

## ◆ 1991年ARM公司成立於英國劍橋

- ◆ 主要出售晶片設計技術的授權。
- ◆ 目前採用ARM技術知識產權（IP）的微處理器，即我們通常所說的ARM微處理器。

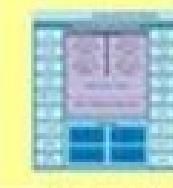
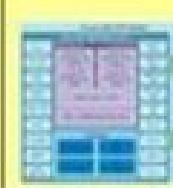
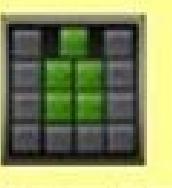
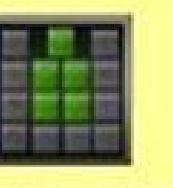
## ◆ ARM現在已遍及

- ◆ 工業控制
- ◆ 消費類電子產品
- ◆ 通信網路系統

## ◆ ARM的微處理器應用約佔

- ◆ 32位元RISC微處理器75%以上的市場比例

# ARM微處理器概述(2/2)

型號	A6	Exynos 4412	Exynos 4412	APQ 8064	Tegra 3	Tegra 3 AP37
代表手機	iPhone5	SAMSUNG Note2	SAMSUNG S3	LG Optimus G	HTC ONE X	HTC ONE X+
CPU						
處理器結構						
ARM架構	Cortex A9	Cortex A9	Cortex A9	Krait A15	Cortex A9	Cortex A9
核心數	2	4	4	4	4+1	4(1.6GHz)+1
常規主頻	1.02~1.29GHz(未諧震)	1.6GHz	1.4GHz	1.5GHz	1.4GHz	1.7GHz
同步/異步	同步	同步	同步	異步	異步	異步
製程工藝	32nm	32nm	32nm	28nm	40nm	40nm
GPU	PowerVR SGX 543 MP3	Mali400	Mali400	Adreno 320	Geforce 416MHz	Geforce 520MHz
L2二級緩存	1MB	1MB	1MB	2MB	1MB	1MB
支援替換	替換	替換	替換	替換	置換	置換?
作業系統	iOS6	Android	Android	Android	Android	Android
DRAM	1GB	2GB	1GB/2GB	2GB	1GB	1GB



# ARM 指令集架構簡介(1/14)

- ◆ 指令集架構是CPU所能看懂的指令格式。
  - ◆ EX：同樣ADD R2,R4,R2 (R2+R4→R2)
    - ◆ MIPS的ISA為0000\_0000\_1000\_0010\_0001\_0000\_0100\_0010
    - ◆ ARM的ISA為1110\_0000\_1000\_0010\_0100\_0000\_0000\_0000
- ◆ 我們主要實現ARM ISA三種TYPE指令
  - ◆ Branch Instructions
    - ◆ 改變program counter(PC)的值，即可作指令的跳躍。
  - ◆ Data Processing Instructions
    - ◆ 根據不同的opcode做對暫存器做運算。
  - ◆ Single Data Transfer instructions
    - ◆ 將記憶體的內容讀到暫存器或將暫存器的內容存到記憶體。



# ARM 指令集架構簡介(2/14)

## ◆ ARM Register

- ◆ ARM 主要擁有16個暫存器 R0~R15個暫存器和CPSR暫存器。
  - ◆ R0~R12通用資料暫存器(32bit)
  - ◆ R13 堆疊暫存器(32bit)
  - ◆ R14 連結暫存器(32bit)
    - 當執行到跳躍指令時，可以用此暫存器來儲存跳躍位置，方便將指令跳回來。
  - ◆ R15(PC) 程式計數暫存器(32bit)
    - 用來儲存指令的位置。
    - PC暫存器為特殊的暫存器
      - 若指令沒有對PC暫存器做寫入的動作，PC暫存器會自行+4。
        - 因此我們沒有將PC暫存器設計到register file內。
  - ◆ CPSR(current processor state register) 處理器狀態暫存器(4bit)
    - 用來儲存現在處理器的狀況，並不在register file內。
    - 在傳統ARM的架構為32bit，但我們這次Final project只會用到CPSR內的**4bit NZCV**因此只需4bit。



# ARM 指令集架構簡介(3/14)

## ◆ Branch Instructions

- ◆ 改變PC(r15)的內容即可跳躍到下一個想要執行的指令。

- ◆  $PC = (PC+4) + (24\text{ bits signed immediate value})^*4$ 
  - $PC+4+imm^*4 \rightarrow PC$

- ◆ ISA :



- ◆ Cond : 條件判斷，根據CPSR的NZCV決定是否執行這道指令。
- ◆ L : Link bit
  - 判斷跳躍時是否要將PC+4的位置存入R14(link register)。
    - L=0 : R14不變； $PC=(PC+4)+immediate^*4$
    - L=1 :  $R14=PC+4$ ； $PC=(PC+4)+immediate^*4$
- ◆ 24 bits Signed Immediate Value : 和PC相加得到跳躍的PC位置。



# ARM 指令集架構簡介(4/14)

## ◆ ARM Condition:

- ◆ ARM 在執行每個指令前都必須經過NZCV暫存器內容來判斷要不要執行這道指令。
  - ◆ EX : ADDEQ R1,R1,R2 (  $R1+R2 \rightarrow R1$  )
    - 在執行ADD前必續判斷Z=1才會去執行ADD這道指令，若Z=0則就不會執行。
- ◆ 透過運算或比較指令去更新negative zero carry overflow(nzcv)
  - ◆ 判斷現在運算的結果是否有negative zero carry overflow(nzcv)
    - 若有則將有的狀態設為1；若無則設為0
  - ◆ EX : CMP R1,R2 ( 假設  $R1=0x00000001$ ,  $R2=0x00000001$  )
    - 運算完的結果為  $R1-R2=0$ ，因為運算結果為0，因此Zero bit就會被設置為1(其餘NCV都沒有發生引此設定為0)。
  - ◆ EX: ADDS R1,R2,R3 (  $R2+R3 \rightarrow R1$  )
    - 假設  $R2$  為  $0xffffffff$ ,  $R3$  為  $0x00000001$
    - ADD後面加S就是代表會根據運算結果更改NZCV；若沒加S則代表不會更改nzcv。
    - 運算結果  $R1 = 0x00000000$   $Z=0$ ，因為有進位  $C=1$  (其餘NV都為0)

# ARM 指令集架構簡介(5/14)

Code	Suffix	Flags	Meaning
0000	EQ	Z=1	Equal
0001	NE	Z=0	Not equal
0010	CS	C=1	Unsigned higher or same
0011	CC	C=0	Unsigned lower
0100	MI	N=1	Negative
0101	PL	N=0	Positive or zero
0110	VS	V=1	Overflow
0111	VC	V=0	No overflow
1000	HI	C = 1 and Z = 0	Unsigned higher
1001	LS	C = 0 or Z = 1	Unsigned lower or same
1010	GE	N = V	Greater or equal
1011	LT	N ≠ V	Less than
1100	GT	Z = 0 and N = V	Greater than
1101	LE	Z = 1 or N ≠ V	Less than or equal
1110	AL	(Ignored)	Always

Condition code



# ARM 指令集架構簡介(6/14)

## ◆ Branch Instructions example:

- ◆ BLEQ #16 (PC+4+16\*4→PC、PC+4→R14)

Cond	101	L	24 bits Signed Immediate Value(offset)
------	-----	---	--

- ◆ 0000\_101\_1\_00000000000000000000000010000
- ◆ 若Z=1 則執行PC=(PC+4)+16\*4、R14 =PC+4

- ◆ BEQ #16 (PC+4+16\*4→PC)

Cond	101	L	24 bits Signed Immediate Value(offset)
------	-----	---	--

- ◆ 0000\_101\_0\_00000000000000000000000010000
- ◆ 若Z=1 則執行PC=(PC+4)+16\*4

- ◆ BGT #-4 (PC+4-4\*4→PC)

Cond	101	L	24 bits Signed Immediate Value(offset)
------	-----	---	--

- ◆ 1100\_101\_0\_111111111111111111111111100
- ◆ 若Z=0 且 N=V 則執行 PC=(PC+4)-4\*4



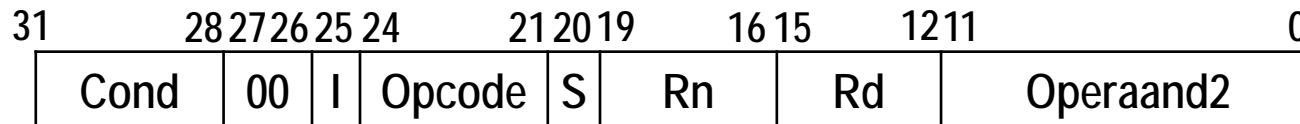
# ARM 指令集架構簡介(7/14)

## ◆ Data Processing Instructions

- ◆ 根據不同的opcode做  $Rd = Rn \text{ op Operand2}$

- ◆  $Rn \text{ op Op2} \rightarrow Rd$

- ◆ ISA:



- ◆ Cond : 條件判斷，根據CPSR的NZCV決定是否執行這段指令。
  - 同上頁的branch指令的condition。
- ◆ I : Immediate bit。
  - $I=0$  : Operand2=register ;  $I=1$  : Operand2=Immediate
- ◆ S : 決定是否根據運算結果更動CPSR的NZCV。
- ◆ Rn : Operand1。
- ◆ Rd : Destination register。
- ◆ Operand2 : 第二運算元

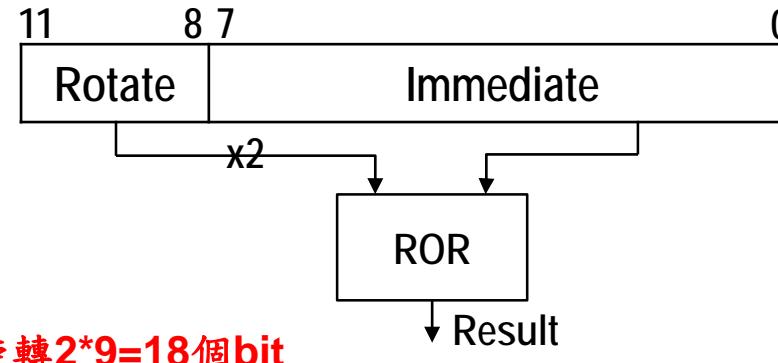
# ARM 指令集架構簡介(8/14)

Assembler Mnemonic	Opcode	Action
AND	0000	Operand1 and Operand2
EOR	0001	Operand1 xor Operand2
SUB	0010	Operand 1 – Operand2
RSB	0011	Operand2 – Operand1
ADD	0100	Operand1 + Operand2
ADC	0101	Operand1+ Operand2 + carry
SBC	0110	Operand1 – Operand2 + carry -1
RSC	0111	Operand2 – Operand1 + carry -1
TST	1000	As AND, but result is not written
TEQ	1001	As XOR, but result is not written
CMP	1010	As SUB, but result is not written
CMN	1011	As ADD, but result is not written
OR	1100	Operand1 or Operand2
MOV	1101	Operand2(Operand1 is ignored)
BIC	1110	Operand1 and not Operand2 (Bit clear)
MVN	1111	not operand2 (operand1 is ignored)

opcode

# ARM 指令集架構簡介(9/14)

- ◆ Immediate bit = 1 → Operand2 = #rot + 8-bit Immediate。
  - ◆ Immed\_8的規範必須式8-bits可以表示，或是以向右偶次數旋轉之後可以用8-bit表式的數字。



■ EX: 12b'1001\_1111\_1111 =

1234 5678 ..... 18  
 32'b0000\_0000\_0011\_1111\_1100\_0000\_0000\_0000

■ EX: 12b'0001\_1001\_1001 =

32'b0100\_0000\_0000\_0000\_0000\_0000\_0010\_0110

■ EX: 12b'1110\_1111\_1111 =

32'b0000\_0000\_0000\_0000\_0000\_1111\_1111\_0000

# ARM 指令集架構簡介(10/14)

◆ Immediate bit = 0 → Operand2 = #shift+ST+0+Rm

◆ 將 Rm 的內容做位移後當作運算元。

11	7 6 5	4 3	0
shift number	ST	0	Rm

■ Shift number :

■ 位移數量的大小5bit不帶符號整數。

■ ST : shift type

- 2'b00 : 邏輯左移(乘法)
- 2'b01 : 邏輯右移
- 2'b10 : 算數右移(除法)
- 2'b11 : 旋轉右移

■ Rm :

■ 被位移的暫存器。

# ARM 指令集架構簡介(11/14)

## ◆ Data Processing Instructions example:

- ◆ ADDS R1,R2,R3 (R2+R3→R1)

Cond	00	I	Op	S	Rn	Rd	SN	St	0	Rm
------	----	---	----	---	----	----	----	----	---	----

■ 1110\_00\_0\_0100\_1\_0010\_0001\_00000\_00\_0\_0011

- ◆ EOR R1,R2,R3>>2 (R2 xor R3→R1)

Cond	00	I	Op	S	Rn	Rd	SN	St	0	Rm
------	----	---	----	---	----	----	----	----	---	----

■ 1110\_00\_0\_0001\_0\_0010\_0001\_00010\_01\_0\_0011

- ◆ ADD R1,R2,0x00000001 (R2+#1→R1)

Cond	00	I	Op	S	Rn	Rd <th>ROR</th> <td>Imm</td>	ROR	Imm
------	----	---	----	---	----	------------------------------	-----	-----

■ 1110\_00\_1\_0100\_0\_0010\_0001\_0000\_00000001

- ◆ ADDNE R1,R2,0x01000000(R2+#64→R1)

Cond	00	I	Op	S	Rn	Rd <th>ROR</th> <td>Imm</td>	ROR	Imm
------	----	---	----	---	----	------------------------------	-----	-----

■ 0001\_00\_1\_0100\_0\_0010\_0001\_0100\_00000001

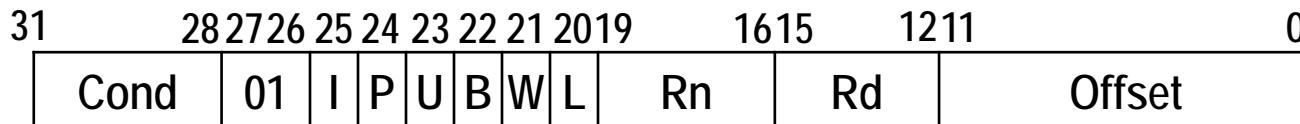
■ 有condition判斷，必須Z=0才執行這道指令。



# ARM 指令集架構簡介(12/14)

## ◆ Single Data Transfer instructions

- ◆ 透過此指令做單一資料的記憶體存取。

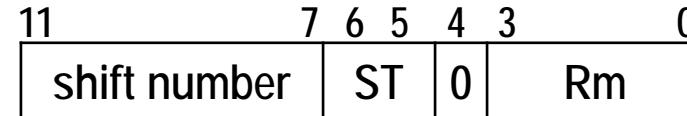


- ◆ Store : Rd → MEM[Rn+Offset] ; Load : MEM[Rn+Offset] → Rd
- ◆ Offset :

- 用於和Base暫存器相加算出address。
- Unsigned 12-bit immediate offset



- Shift operation to register



- ◆ Rd : Destination register
  - 想要儲存或讀取的暫存器。

# ARM 指令集架構簡介(13/14)

- ◆ Rn : Base register
  - 用於和Offset相加算出address
- ◆ L : Load/store bit
  - L=0 : store to memory ; L=1 : load to register
- ◆ W : write-back bit (**Our final project always is '0'**)
  - W=0 : No write-back ; W=1 : write address to base register
- ◆ B : Byte/Word bit (**Our final project always is '0'**)
  - B=0 : transfer word quantity ; B=1 : transfer byte quantity
- ◆ U : Up/Down bit
  - U=0 : subtract offset from base ; U=1 : add offset to base
- ◆ P : Pre/Post indexing bit (**Our final project always is '1'**)
  - P=0 : Post ,add offset after transfer ;
  - P=1 : Pre, add offset before transfer
- ◆ I : Immediate bit
  - I=1 : Offset=register ; I=0 : Offset=Immediate

# ARM 指令集架構簡介(14/14)

## ◆ Single Data Transfer Instructions example:

- ◆ ST R5,[R4+R3] (R5→MEM[R4+R3])

Cond	01	I	P	U	B	W	L	Rn	Rd	SN	St	0	Rm
■	1110_01_1_1_1_0_0_0_0100_0101_00000_00_0_0011												

- ◆ LD R5,[R4+R3<<2] (MEM[R4+R3<<2]→R5)

Cond	01	I	P	U	B	W	L	Rn	Rd	SN	St	0	Rm
■	1110_01_1_1_1_0_0_1_0100_0101_00010_00_0_0011												

- ◆ LD R5,[R4+#8] (MEM[R4+8]→R5)

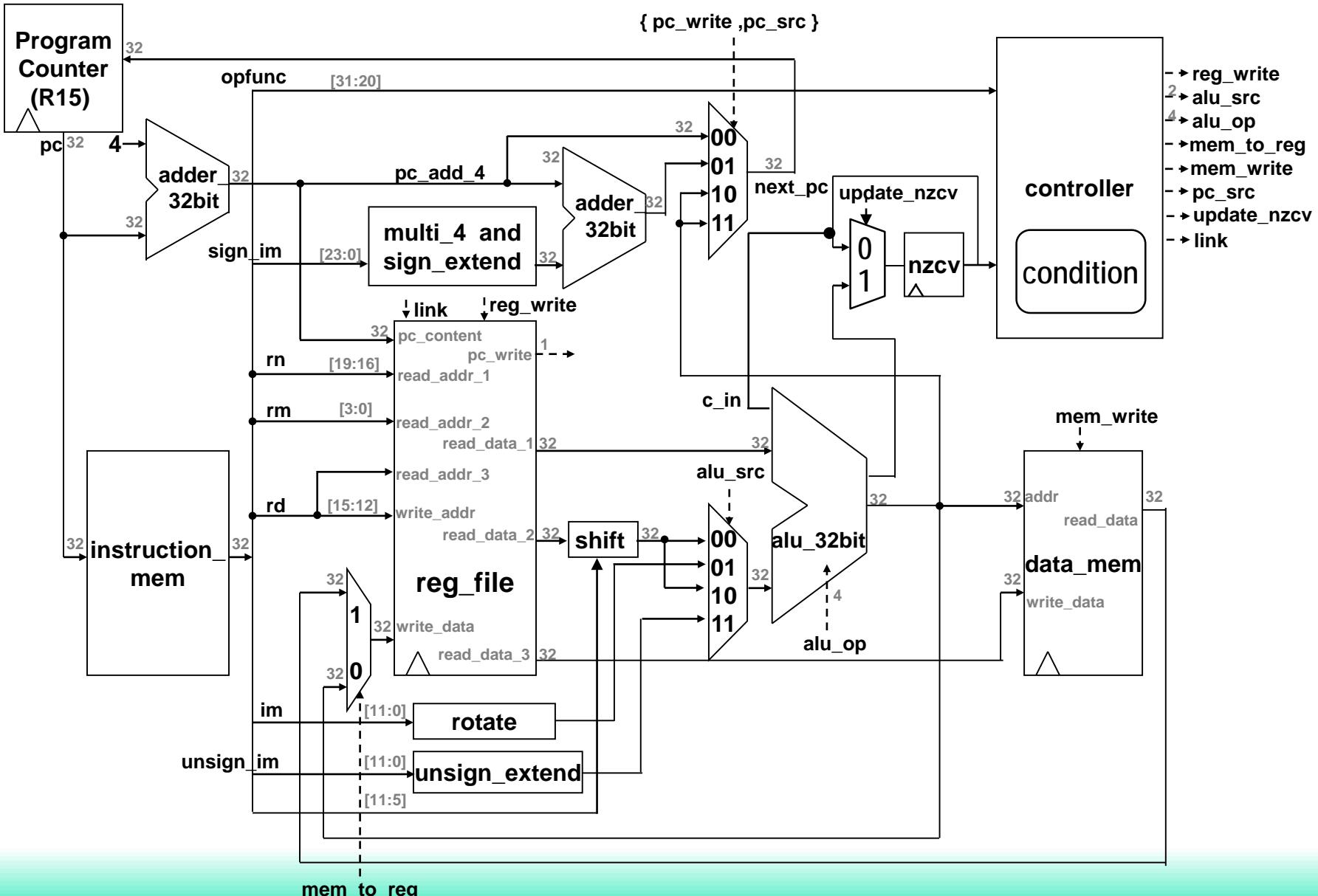
Cond	01	I	P	U	B	W	L	Rn	Rd	Immediate			
■	1110_01_0_1_1_0_0_1_0100_0101_000000001000												

- ◆ LDGT R5,[R4-#8] (MEM[R4-8]→R5)

Cond	01	I	P	U	B	W	L	Rn	Rd	Immediate			
■	1100_01_0_1_0_0_0_1_0100_0101_000000001000												

■ 有condition判斷，必須Z=0且N=V才執行這道指令。

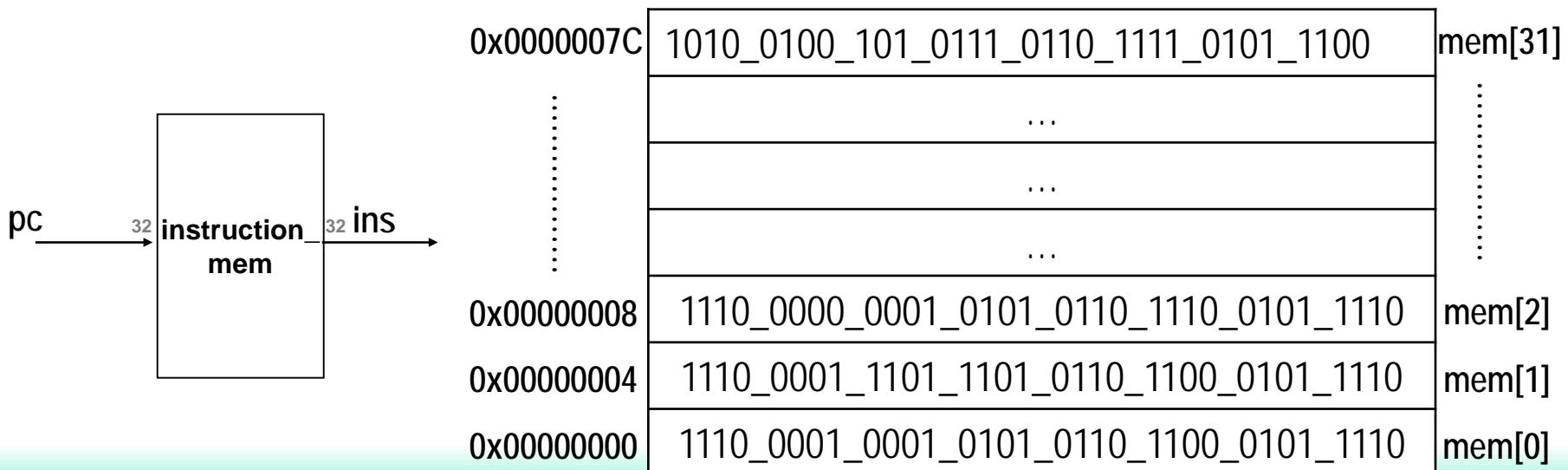
# Single cycle ARM implementation(1/25)



# Single cycle ARM implementation(2/25)

## ◆ Instruction memory :

- ◆ 組合邏輯電路
- ◆ 輸入
  - 32\_bit : pc 。
- ◆ 輸出
  - 32\_bit : ins
- ◆ 合成面積考量，我們只實現存放32個指令的記憶體。
- ◆ 將pc所選擇的記憶體位址中的一個word(32bit)大小的指令給讀出



# Single cycle ARM implementation(3/25)

請務必使用 ins\_mem 做為名子，這樣 testbench 才不會錯。

```
module ins_mem(pc, ins);
  input [31:0] pc;
  output [31:0] ins;
  parameter INS_MEM_SIZE = 32;
  reg [31:0] mem [INS_MEM_SIZE-1:0];
  assign ins = mem [ pc[31:2] ];
endmodule
```

memory 的大小為 32 個 word (1\_word=32\_bit=4\_byte)

宣告 reg 二為陣列，大小為 32bit 的 32 個 reg，  
 請務必使用 mem 做為名子，這樣 testbench 才不會錯。

因為我們 ins\_mem 是以 word(4\_byte) 為一單位，所以取 pc[31:2]，  
 EX:PC=0000\_0000\_0000\_0000\_0000\_0000\_0100 (PC=4)  
 則抓取 mem[1] 的指令；  
 EX:PC=0000\_0000\_0000\_0000\_0000\_0000\_1100 (PC=12)  
 則抓取 mem[3] 的指令。



# Single cycle ARM implementation(4/25)

ISE Project Navigator (O.61xd) - D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Final project\ARM\ARM.xise - [ins\_mem.v]

File Edit View Project Source Process Tools Window Layout Help

Design View: Implementation Simulation

Hierarchy

- ARM
  - xc6slx25-3ftg256
  - ARM (ARM.v)
    - \_ins\_mem - ins\_mem (ins\_n)
    - \_register\_file - register\_file (re)
    - \_sign\_extent - sign\_extent (sig)
    - \_rotate - rotate (rotate.v)
    - \_unsigned\_extend - unsigned\_exten

No Processes Running

Processes: \_ins\_mem - ins\_mem

- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
  - View RTL Schematic
  - View Technology Schematic

Start Design Files Libraries Design Summary (Synthesized) ARM.v register\_file.v ins\_mem.v

```

11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 /////////////////////////////////
21 module ins_mem(pc, ins);
22
23   input [31:0] pc;
24   output [31:0] ins;
25
26   parameter INS_MEM_SIZE = 32;
27
28   reg [31:0] mem [INS_MEM_SIZE-1:0];
29
30   assign ins = mem [ pc[31:2] ];
31
32 endmodule
33

```

Warnings

[WARNING]:Xst:647 - Input <pc<31:7>> is never used. This port will be preserved and left unconnected if it belongs to a top-level component.
 [WARNING]:Xst:647 - Input <pc<1:0>> is never used. This port will be preserved and left unconnected if it belongs to a top-level component.
 [WARNING]:Xst:653 - Signal <mem> is used but never assigned. This sourceless signal will be automatically connected to value 0.

**Waring Xst:647 告知我們 PC<31:7>、PC<1:0>沒有被使用。**

**Wargin Xst:653 告知我們雖然有宣告mem但是並完全沒有給內容，這我們在testbench 才會給值。**

Warnings Errors

Ln 21 Col 25 Verilog

# Single cycle ARM implementation(5/25)

## ◆ Data memory :

### ◆ 循序邏輯電路

#### ◆ 輸入：

- 1\_bit : clk, rst, mem\_write
- 32\_bit : addr, write\_data

#### ◆ 輸出：

- 32\_bit : read\_data

◆ 合成面積考量，我們只實現存放64個資料的記憶體。

### ◆ 將資料寫入記憶體時(store word)：

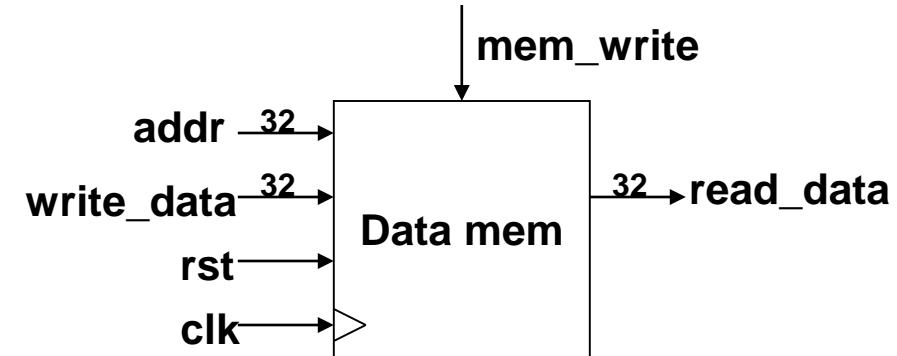
#### ◆ 動作如同序向電路。

◆ 當clk正緣觸發且mem\_write =1時，將一個word大小的資料(write\_data)給寫入addr所選擇的記憶體位址

### ◆ 將資料讀出記憶體時(load word)：

#### ◆ 動作如同組合電路。

◆ 將addr所選擇的記憶體位址中的一個word大小的資料給讀出(read\_data)。



# Single cycle ARM implementation(6/25)

請務必使用data\_mem做為名子，這樣testbench才不會錯。

```

module data_mem(clk,rst,addr,write_data,mem_write,read_data
);
  input clk,rst,mem_write;
  input [31:0]addr,write_data;
  output [31:0]read_data;

  parameter DATA MEM SIZE = 64;

  //register
  reg [31:0] mem [DATA_MEM_SIZE-1:0];

  integer i;
  assign read data = mem[ addr[31:2] ];

  always@(posedge clk or posedge rst)
  begin
    if( rst == 1'b1 )
      for( i=0 ; i<DATA_MEM_SIZE ; i=i+1)
        mem[i] <= 0;
    else if ( mem_write == 1'b1)
      mem[ addr[31:2] ] <= write_data;
  end

endmodule

```

memory的大小為64個word (1\_word=32\_bit=4\_byte)

宣告reg二為陣列，大小為32bit的64個reg

請務必使用mem做為名子，這樣testbench才不會錯。

將addr所選擇的記憶體位址中的一個word大小的資料給讀出

rst訊號對記憶體做清除

mem\_write訊號對ddr所選擇的記憶體位址做寫入

# Single cycle ARM implementation(7/25)

ISE Project Navigator (O.61xd) - D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Final project\ARM\ARM.xise - [data\_mem.v]

File Edit View Project Source Process Tools Window Layout Help

Design Implementation Simulation

Hierarchy

```

21 module data_mem(clk,rst,addr,write_data,mem_write,read_data
22 );
23   input clk,rst,mem_write;
24   input [31:0]addr,write_data;
25   output [31:0]read_data;
26
27 parameter DATA_MEM_SIZE = 64;
28
29 //register
30 reg [31:0] mem [DATA_MEM_SIZE-1:0];
31
32
33 integer i;
34
35 assign read_data = mem[ addr[31:2] ];
36
37 always@(posedge clk or posedge rst)
38 begin
39   if( rst == 1'b1 )
40     for( i=0 ; i<DATA_MEM_SIZE ; i=i+1)
41       mem[i] <= 0;
42   else if ( mem_write == 1'b1)
43     mem[ addr[31:2] ] <= write_data;

```

No Processes Running

Processes: \_data\_mem - data\_mem

- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
  - View RTL Schematic
  - View Technology Schematic

Start Design Files Libraries

Design Summary (Synthesized) ARM.v register\_file.v ins\_mem.v data\_mem.v

Warnings

**WARNING:Xst:647** - Input <addr<1:0> is never used. This port will be preserved and left unconnected if it belongs to a top-level module.

**Waring Xst:647** 告知我們 PC<31:7>、PC<1:0>沒有被使用。

**Wargin Xst:653** 告知我們雖然有宣告mem但是並完全沒有給內容，這我們在testbench 才會給值。

Ln 16 Col 14 Verilog

# Single cycle ARM implementation(8/25)

## ◆ register file :

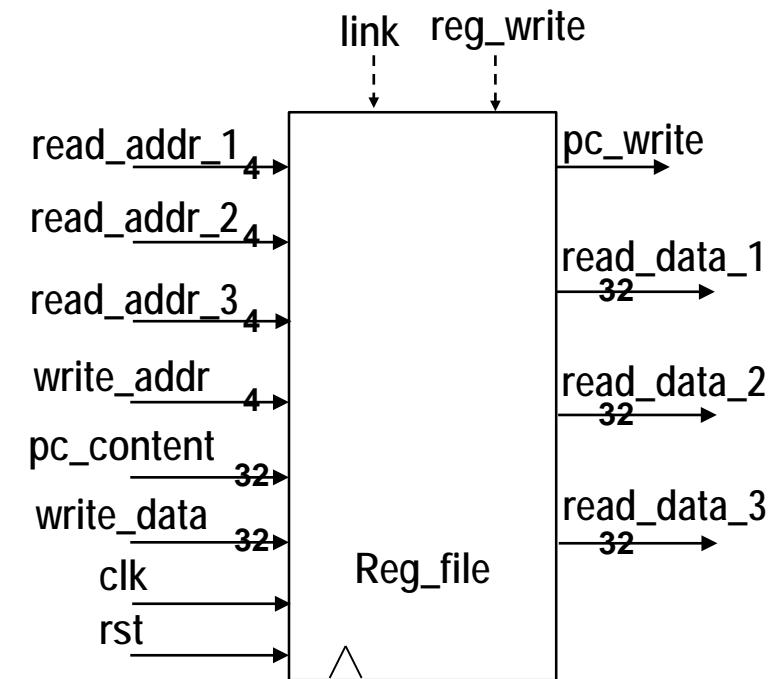
◆ 循序邏輯電路

◆ 輸入：

- ◆ 1\_bit : clk, rst, reg\_write, link
- ◆ 4\_bit : read\_addr\_1, read\_addr\_2, read\_addr\_3, write\_addr
- ◆ 32\_bit : write\_data, pc\_content

◆ 輸出：

- ◆ 1\_bit : pc\_wirte
- ◆ 32\_bit : read\_data\_1, read\_data\_2, read\_data\_3



# Single cycle ARM implementation(9/25)

## ◆ register file 讀取行為：

- ◆ 將read\_addr\_1所選擇的暫存器中的資料讀出到read\_data\_1。
  - read\_addr\_1 可能為R0~R15，
    - R15為特別的PC暫存器，因此我們沒有設計到register file裡，
      - 若要讀取R15 則將pc\_content的值輸出到read\_data\_1
- ◆ 將read\_addr\_2所選擇的暫存器中的資料讀出到read\_data\_2。
  - read\_addr\_2 可能為R0~R15，
    - R15為特別的PC暫存器，因此我們沒有設計到register file裡
      - 若要讀取R15 則將pc\_content的值輸出到read\_data\_2
- ◆ 將read\_addr\_3所選擇的暫存器中的資料讀出到read\_data\_3。
  - read\_addr\_3 可能為R0~R15，
    - R15為特別的PC暫存器，因此我們沒有設計到register file裡，
      - 若要讀取R15 則將pc\_content的值輸出到read\_data\_3



# Single cycle ARM implementation(10/25)

## ◆ register file 寫入行為：

### ◆ reg\_write 訊號：

- 當 $\text{reg\_write} = 1$ 時，將 $\text{write\_data}$ 寫入 $\text{write\_addr}$ 所選擇的暫存器中。
  - $\text{write\_addr}$  可能為R0~R15，
    - R15為特別的PC暫存器，因此我們沒有設計到register file裡，
      - 若要寫入R15 則將 $\text{pc\_write}=1$ 。(  $\text{pc\_write}$ 只是一條線，用組合電路描述)
    - 當 $\text{reg\_write}=0$ 時，則不做寫入動作。

### ◆ link 訊號：

- 當 $\text{link} = 1$ 時，將 $\text{pc\_content}$ 寫入R14。
  - 當 $\text{link} = 0$ 時，則不做寫入動作。

# Single cycle ARM implementation(11/25)

## ◆ multi\_4 and sign extend

◆ 組合邏輯電路

◆ 輸入：

- ◆ 24\_bit : sign\_immediate\_in

◆ 輸出：

- ◆ 32\_bit : sign\_extend\_immediate\_out

◆ 將輸入的值乘4(左移兩bit)並且做有號數擴展到32bit

- ◆ EX : in = 0000\_0000\_0000\_0000\_0000\_0001 (#1)

out = 0000\_0000\_0000\_0000\_0000\_0000\_0100 (#4)

- ◆ EX : in = 1111\_1111\_1111\_1111\_1111\_1100 (-4)

out = 1111\_1111\_1111\_1111\_11111\_1111\_111\_0000 (-16)



# Single cycle ARM implementation(12/25)

## ◆ rorotate

- ◆ 組合邏輯電路

- ◆ 輸入：

- ◆ 12\_bit : immediate\_in

- ◆ 輸出：

- ◆ 32\_bit : rorate\_immediate\_out

- ◆ 將immediate\_in[7:0]向右旋轉immediate\_in[11:8]\*2後輸出

- ◆ Ex : in = 0001\_1001\_1001 (將1001\_1001往右旋轉2個bit)

- out = 0100\_0000\_0000\_0000\_0000\_0000\_0010\_0110

- ◆ Ex : in = 1110\_1111\_1111 (將1111\_1111往右旋轉28個bit)

- out = 0000\_0000\_0000\_0000\_0000\_1111\_1111\_0000



# Single cycle ARM implementation(13/25)

## ◆ shift

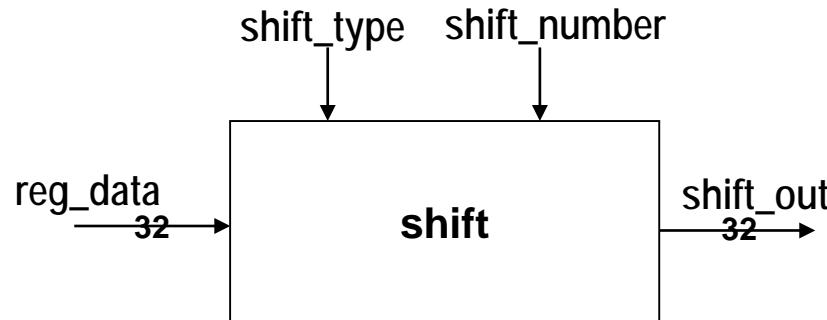
- ◆ 組合邏輯電路

- ◆ 輸入：

- ◆ 2\_bit : shift\_type (位移的種類)
- ◆ 5\_bit : shift\_number (位移的bit數)
- ◆ 32\_bit : reg\_data (暫存器讀出來的值)

- ◆ 輸出：

- ◆ 32\_bit : shift\_out





# Single cycle ARM implementation(14/25)

◆ 根據shift\_type做四種位移的運算，根據shift\_number當做要位移的量。

## ◆ shift\_type

- 2'b00:邏輯左移（即往左移，右邊補零，有乘法效果）
- 2'b01:邏輯右移（即往右移，左邊補零）
- 2'b10:算數右移（即往右移，左邊補signed\_bit，有除法效果）
- 2'b11:旋轉右移（即往右移，左邊補右移過去的值）

## ◆ Ex: shift\_type = 2'b00 , shift\_number = 00010, (邏輯左移2bit)

in = 0000\_000\_0000\_0000\_0000\_0000\_0111\_0000 (#112)

out = 0000\_000\_0000\_0000\_0000\_0001\_1100\_0000 (#448)

## ◆ Ex: shift\_type = 2'b10 , shift\_number = 00100, (算數右移4bit)

in = 1111\_1111\_1111\_1111\_1111\_1111\_1100\_0000 (#-64)

out = 1111\_1111\_1111\_1111\_1111\_1111\_1111\_1100 (#-4)

# Single cycle ARM implementation(15/25)

## ◆ unsign\_extend

◆ 組合邏輯電路

◆ 輸入：

- ◆ 12 bit : unsign\_immediate\_in

◆ 輸出：

- ◆ 32 bit : unsign\_extend\_immediate\_out

◆ 將輸入做無號數擴展到32bit

- ◆ EX: in = 1110\_1100\_1000

out = 0000\_0000\_0000\_0000\_0000\_1110\_1100\_1000

- ◆ EX: in = 0001\_1010\_1011

out = 0000\_0000\_0000\_0000\_0000\_0001\_1010\_1011



# Single cycle ARM implementation(16/25)

## ◆ alu (arithmetic logic unit)

### ◆ 組合邏輯電路

#### ◆ 輸入：

- 32\_bit : source\_1, source\_2
- 4\_bit : alu\_op
- 1\_bit : c\_in

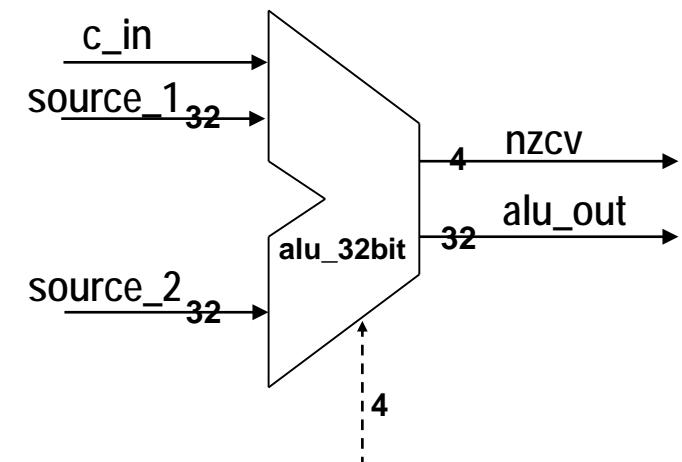
#### ◆ 輸出：

- 4\_bit : nzvc( negative zero overflow carry )
- 32\_bit : alu\_out

◆ 根據 alu\_op 做  $alu\_out = source\_1 \text{ op } source\_2$  的運算。

◆ 根據 alu\_out 改變 nzcv 。

- ◆  $alu\_out == 0$  則  $z=1$  ;  $alu\_out != 0$  則  $z=1$
- ◆  $alu\_out$  = 負數 則  $n=1$  ;  $alu\_out$  = 正數 則  $n=0$
- ◆  $alu\_out$  有進位或無借位 則  $c=1$  ;  $alu\_out$  沒進位或有借位 則  $c=0$
- ◆  $alu\_out$  有溢位 則  $v=1$  ;  $alu\_out$  沒溢位 則  $v=0$



# Single cycle ARM implementation(17/25)

## ◆ alu\_opcode

alu_pcode	Action
0000	Operand1 and Operand2
0001	Operand1 xor Operand2
0010	Operand 1 – Operand2
0011	Operand2 – Operand1
0100	Operand1 + Operand2
0101	Operand1 + Operand2 + carry
0110	Operand1 – Operand2 + carry -1
0111	Operand2 – Operand1 + carry -1
1000	As AND, but result is not written
1001	As XOR, but result is not written
1010	As SUB, but result is not written
1011	As ADD, but result is not written
1100	Operand1 or Operand2
1101	Operand2(Operand1 is ignored)
1110	Operand1 and not Operand2 (Bit clear)
1111	not operand2 (operand1 is ignored)



# Single cycle ARM implementation(18/25)

- ◆ EX : alu\_op = 0000(and) , c\_in=0

source\_1 = 1000\_ 0000\_ 0110\_ 0010\_ 0110\_ 0000\_ 0000\_ 0110

source\_2 = 1100\_ 1110\_ 0111\_ 0110\_ 0111\_ 0000\_ 0110\_ 1111

alu\_out = 1000\_ 0000\_ 0110\_ 0010\_ 0110\_ 0000\_ 0000\_ 0110

nzcv = 1000

(輸出結果為負數 n=1 , C=0 且 V=0 邏輯運算沒進位或溢位)

- ◆ EX : alu\_op = 0100(add) , c\_in=0

source\_1 = 1111\_ 1111\_ 1111\_ 1111\_ 1111\_ 1111\_ 1111\_ 1111

source\_2 = 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0001

alu\_out = 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000

nzcv = 0110

(若無號數運算是 $4294967295+1=0$ ，有進位 c=1；

若有號數運算是 $-1+1=0$ ，沒有 overflow v=0；

alu\_out 結果為0，zero=0 )



# Single cycle ARM implementation(19/25)

- ◆ EX : alu\_op = 0010(sub) , c\_in = 0

source\_1 = 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000

source\_2 = 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0001

alu\_out = 1111\_ 1111\_ 1111\_ 1111\_ 1111\_ 1111\_ 1111\_ 1111

nzcv = 1000

( 若無號數運算是 $0-1=-1$ ，有借位  $c=0$ ；

若有號數運算是 $0-1=-1$ ，沒有overflow  $v=0$ ； )

- ◆ EX : alu\_op = 0100(add) , c\_in=0

source\_1 = 1000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000

source\_2 = 1000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000

alu\_out = 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000\_ 0000

nzcv = 0111

( 若無號數運算是 $2147483648+2147483648=0$ ，有進位  $c=1$ ；

若有號數運算是 $-2147483648-2147483648=0$ ，有overflow  $v=1$ ；

alu\_out 結果為0，zero=1 )

# Single cycle ARM implementation(20/25)

## ◆ controller

- ◆ 組合邏輯電路

- ◆ 輸入:

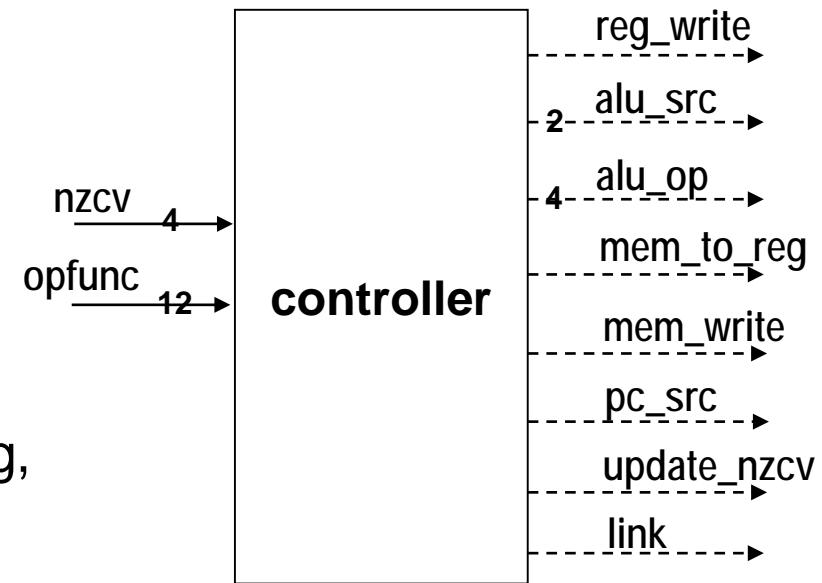
- ◆ 4\_bit : nzcv
- ◆ 12\_bit : opfunc

- ◆ 輸出:

- ◆ 1\_bit : reg\_write, mem\_to\_reg,  
mem\_write, pc\_src,  
update\_nzcv, link

- ◆ 2\_bit : alu\_src
- ◆ 4\_bit : alu\_op

- ◆ 根據輸入的opfunc判斷要做輸出的控制訊號





# Single cycle ARM implementation(21/25)

- ◆ 根據opfunc[11:8]比較nzcv，條件判斷成立才會執行指令。

opfunc[11:8]	條件判斷
0000	Z=1
0001	Z=0
0010	C=1
0011	C=0
0100	N=1
0101	N=0
0110	V=1
0111	V=0
1000	C=1 and Z=0
1001	C=0 or Z=1
1010	N = V
1011	N ≠ V
1100	Z=0 and N = V
1101	Z set or N ≠ V
1110	(Ignored)

# Single cycle ARM implementation(22/25)

## ◆ 條件判斷失敗：

- `reg_write = 1'b0 ; alu_src = 2'b00 ; alu_op = 4'b0000 ;`
- `mem_to_reg = 1'b0 ; mem_write = 1'b0 ; pc_src = 1'b0 ;`
- `update_nzcv = 1'b0 ; link = 0;`

## ◆ 條件判斷成功：

- 根據 `opfunc[7:0]` 判斷是何種type的指令輸出的控制訊號。
  - `Branch_type :`
    - `reg_write = 1'b0 ; alu_src = 2'b00 ; alu_op = 4'b0000 ;`
    - `mem_to_reg = 1'b0 ; mem_write = 1'b0 ; pc_src = 1'b1 ;`
    - `update_nzcv = 1'b0 ; link = opfunc[4];`
  - `Data_process_type :`
    - `Opfunc[4:1] = 1000 或 1001 或 1010 或 1011 , 則 Reg_write = 1'b0;`
    - `Opfunc[4:1] != 1000 或 1001 或 1010 或 1011 , 則 Reg_write = 1'b1;`



# Single cycle ARM implementation(23/25)

- opfunc[5] == 0 , 則 alu\_src = 2'b00
- opfunc[5] == 1 , 則 alu\_src = 2'b01
- alu\_op = opfunc[4:1] ; mem\_to\_reg = 1'b0 ; mem\_write = 1'b0 ;
- pc\_src = 1'b0 ; update\_nzcv = opfunc[0]; link = 1'b0;
- Data transfer type:
  - reg\_write = opfunc[0] ;
  - opfunc[5] == 1'b1 , 則 alu\_src = 2'b10;
  - opfunc[5] == 1'b0 , 則 alu\_src = 2'b11;
  - opfunc[3] == 1'b1 , 則 alu\_op = 4'b0100 ;
  - opfunc[3] == 1'b0 , 則 alu\_op = 4'b0010 ;
  - mem\_to\_reg = 1'b1;
  - mem\_write = ~ opfunc[0] ; pc\_src = 1'b0 ; update\_nzcv = 1'b0 ; link = 1'b0;

# Single cycle ARM implementation(24/25)

## ◆ ARM top module:

- ◆ 除了上述9個 module 外

- ◆ 還需要兩個 adder

  - ◆ assign pc\_add\_4 = pc+32'd4;

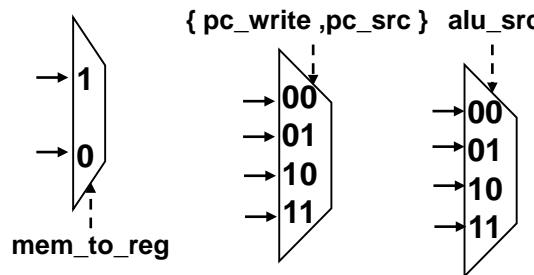
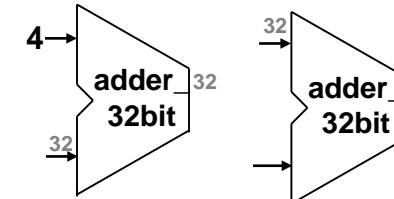
  - ◆ assign pc\_branch = sign\_extend\_out + pc\_add\_4;

- ◆ 三個 mux

  - ◆ assign reg\_write\_data = mem\_to\_reg?(mem\_read\_data):(alu\_out);

  - ◆ assign pc\_next = pc\_write? ( pc\_src? (alu\_out):(alu\_out) ):  
( pc\_src? (pc\_branch):(pc\_add\_4) );

  - ◆ assign alu\_operation\_2 = alu\_src[1]?  
( alu\_src[0]? unsign\_extend\_out):(shift\_out) ):  
( alu\_src[0]? rorate\_out):(shift\_out) );



# Single cycle ARM implementation(25/25)

ISE Project Navigator (O.61xd) - D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Final project\ARM\ARM.xise - [ARM.v]

File Edit View Project Source Process Tools Window Layout Help

Design View: Implementation Simulation

Hierarchy

- ARM
  - xc6slx25-3ftg256
    - ARM (ARM.v)
      - \_ins\_mem - ins\_mem (ins\_mem.v)
      - \_register\_file - register\_file (register\_file.v)
      - \_sign\_extend - sign\_extend (sign\_extend.v)
      - \_rotate - rotate (rotate.v)

No Processes Running

Processes: ARM

- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- View RTL Schematic

Start Design Files Libraries

Design Summary (Synthesized) ARM.v register\_file.v ins\_mem.v data\_mem.v

Warnings

```

WARNING:Xst:2972 - "d:/copy/ncku/ncku_soc/digital logic experiments/logic desing 104/final project/arm/arm.v" line 46. All
WARNING:Xst:2972 - "d:/copy/ncku/ncku_soc/digital logic experiments/logic desing 104/final project/arm/arm.v" line 47. All
WARNING:Xst:2972 - "d:/copy/ncku/ncku_soc/digital logic experiments/logic desing 104/final project/arm/arm.v" line 51. All
WARNING:Xst:2972 - "d:/copy/ncku/ncku_soc/digital logic experiments/logic desing 104/final project/arm/arm.v" line 52. All
WARNING:Xst:2972 - "d:/copy/ncku/ncku_soc/digital logic experiments/logic desing 104/final project/arm/arm.v" line 53. All
WARNING:Xst:2972 - "d:/copy/ncku/ncku_soc/digital logic experiments/logic desing 104/final project/arm/arm.v" line 54. All
WARNING:Xst:2972 - "d:/copy/ncku/ncku_soc/digital logic experiments/logic desing 104/final project/arm/arm.v" line 55. All
WARNING:Xst:2972 - "d:/copy/ncku/ncku_soc/digital logic experiments/logic desing 104/final project/arm/arm.v" line 56. All
WARNING:Xst:2972 - "d:/copy/ncku/ncku_soc/digital logic experiments/logic desing 104/final project/arm/arm.v" line 58. All

```

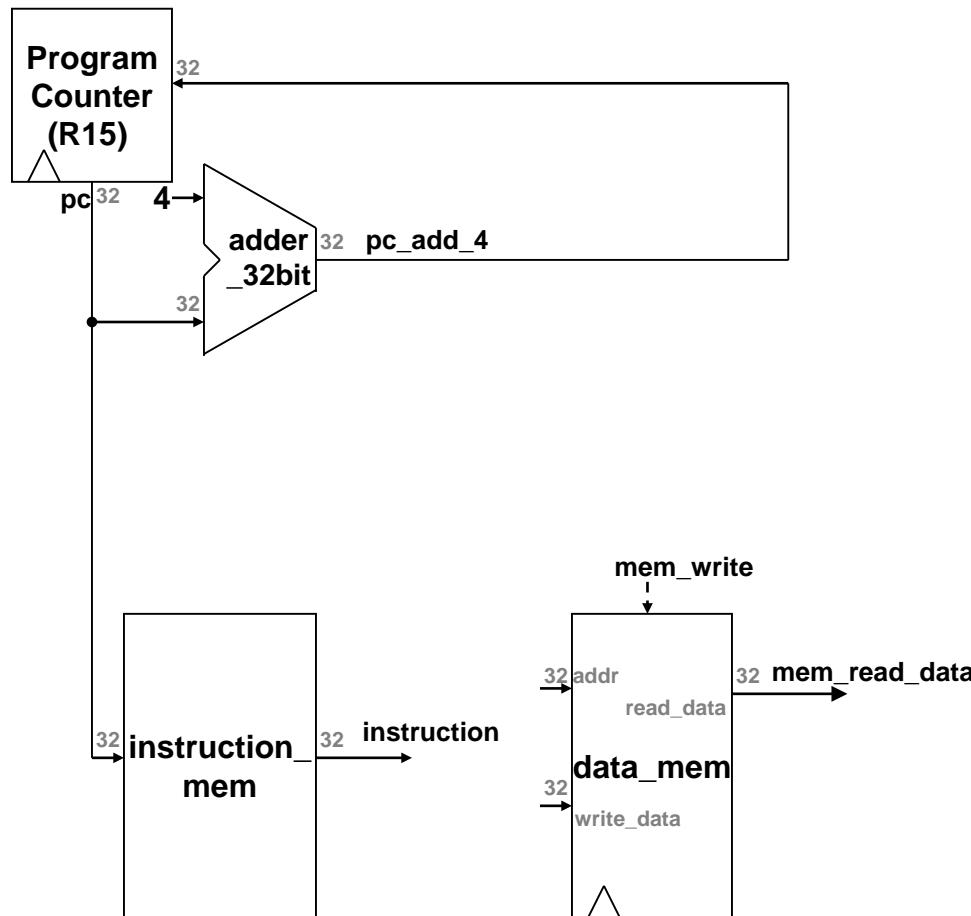
Waring 2972 : 告知我們使用的module和我們的電路輸出沒有關係，因此幫我們移除掉，  
但我們依然可以用testbench去跑模擬。

Ln 30 Col 31 | Verilog  
SOC & ASIC Lab

# Final project code(1/3)

- ◆ 在FTP上面的Final project code下載testbench 當
  - ◆ ARM：請務必使用此檔案做top module。
  - ◆ tb\_ARM\_1~4.v：
    - ◆ test\_bench\_1：記憶體測試
    - ◆ test\_bench\_2：指令測試
    - ◆ test\_bench\_3：bubble sort程式
    - ◆ test\_bench\_4：指令測試
  - ◆ arm\_tb\_mem\_data\_1~4.txt：
    - ◆ data\_memory的內容
  - ◆ arm\_tb\_program\_1~4.txt：
    - ◆ ins\_memory的內容(即是指令)
  - ◆ arm\_tb\_answer\_1~4.txt：
    - ◆ testbench的答案(data\_memory執行完後的內容)

# Final project code(2/3)



ARM.v 目前已經實現的電路  
(打好inst\_mem和data\_mem就可以直接跑testbench1)

# Final project code(3/3)

ISE Project Navigator (O.61xd) - D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Final project\ARM\ARM.xise - [ARM2.v\*]

File Edit View Project Source Process Tools Window Layout Help

Design View: Implementation Simulation

Hierarchy

- ARM
  - xc6slx25-3ftg256
    - ARM (ARM.v)
      - \_ins\_mem - ins\_mem (ins\_mem.v)
      - \_register\_file - register\_file (register\_file.v)
      - \_sign\_extent - sign\_extent (sign\_extent.v)
      - \_rorate - rorate (rorate.v)
      - \_unsigned\_extend - unsigned\_extend (unsigned\_extend.v)
      - \_shift - shift (shift.v)
      - \_alu - alu (alu.v)
      - \_control - control (control.v)
      - \_data\_mem - data\_mem (data\_mem.v)
    - ARM2 (ARM2.v)
      - \_ins\_mem - ins\_mem (ins\_mem.v)
      - \_data\_mem - data\_mem (data\_mem.v)

No Processes Running

Processes: ARM2

- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
  - View RTL Schematic
  - View Technology Schematic
  - Check Syntax
  - Generate Post-Synthesis Simulation ...
- Implement Design
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

```

13 // Dependencies:
14 // Revision:
15 //
16 // Revision 0.01 - File Created
17 // Additional Comments:
18 //
19 //
20 /////////////////////////////////////////////////
21 module ARM(clk,rst
22 );
23   input clk,rst;
24
25   //register
26   reg [31:0]pc;
27
28   //wire
29   wire [31:0]pc_add_4, mem_read_data, instruction;
30
31   //adder
32   assign pc_add_4 = pc+32'd4;
33
34   ins_mem _ins_mem(.pc(pc), .ins(instruction));
35   data_mem _data_mem(.clk(clk), .rst(rst), .addr(32'b0), .write_data(32'b0), .mem_write(1'b0),
36   .read_data(mem_read_data));
37
38   always@(posedge clk or posedge rst)
39   begin
40     if( rst == 1'b1 )
41       pc = 32'd0;
42     else
43       pc = pc_add_4;
44   end
45
46
47 endmodule
  
```

Design Summary (Synthesized) ARM.v ARM2.v\* tb\_ARM1.v

Warnings

Waring HDLCompiler1127: 告知我們ins\_mem和data\_mem的輸出腳沒接東西

WARNING:HDLCompiler:1127 - "D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Final project\ARM\ARM2.v" Line 34: Assignment to instruction i

WARNING:HDLCompiler:1127 - "D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Final project\ARM\ARM2.v" Line 36: Assignment to mem\_read\_data

WARNING:Xst:2972 - "d:/copy/ncku/ncku\_soc/digital logic experiments/logic desing 104/final project/arm/arm2.v" line 34. All outputs of instance <\_ins\_mem>

WARNING:Xst:2972 - "d:/copy/ncku/ncku\_soc/digital logic experiments/logic desing 104/final project/arm/arm2.v" line 35. All outputs of instance <\_data\_mem>

Waring 2972 : 告知我們使用的module和我們的電路輸出沒有關係，因此幫我們移除掉，  
但我們依然可以用testbench去跑模擬。

# Testbench for ARM(1/7)

ISE Project Navigator (0.61xd) - D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Final project\ARM\ARM.xise - [tb\_ARM\_3.v]

File Edit View Project Source Process Tools Window Layout Help

Design View: Implementation Simulation Behavioral

Hierarchy

- ARM
  - xc6slx25-3ftg256
    - tb\_ARM\_1 (tb\_ARM\_1.v)
    - tb\_ARM\_2 (tb\_ARM\_2.v)
    - tb\_ARM\_3 (tb\_ARM\_3.v) **selected**
    - tb\_ARM\_4 (tb\_ARM\_4.v)

```

1 `timescale 1ns / 1ps
2 `define CYCLE 5
3 `define PROGRAM "arm_tb_program_3.txt"
4 `define MEM_DATA "arm_tb_mem_data_3.txt"
5 `define ANSWER "arm_tb_answer_3.txt"

6
7 /////////////////
8 // Company:
9 // Engineer:
10 //
11 // Create Date: 15:38:26 04/25/2014
12 // Design Name: ARM
13 // Module Name: D:/Copy/NCKU/_SoC/Digital Logic Ex]
14 // Project Name: ARM
15     integer error, i;
16     reg [31:0]tb_answer[DATA_MEM_SIZE-1: 0];

17
18
19     always #(`CYCLE) clk=~clk;

20
21     initial begin
22         // Initialize Inputs
23         clk = 0; $readmemb(`PROGRAM, uut._ins_mem.mem);
24         rst = 1; $readmemh(`MEM_DATA, uut._data_mem.mem);
25         error = 0; $readmemh(`ANSWER, tb_answer);
26         #(`CYCLE*2) rst = 0;
27
28
29
30
31
32
33
34
35
36
37
38
39
39
40
41
42
43
44
45
46
47
48
49
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
62
63
63
64
64
65
65
66
66
67
67
68
68
69
69
70
70
71
71
72
72
73
73
74
74
75
75
76
76
77
77
78
78
79
79
80
80
81
81
82
82
83
83
84
84
85
85
86
86
87
87
88
88
89
89
90
90
91
91
92
92
93
93
94
94
95
95
96
96
97
97
98
98
99
99
100
100
101
101
102
102
103
103
104
104
105
105
106
106
107
107
108
108
109
109
110
110
111
111
112
112
113
113
114
114
115
115
116
116
117
117
118
118
119
119
120
120
121
121
122
122
123
123
124
124
125
125
126
126
127
127
128
128
129
129
130
130
131
131
132
132
133
133
134
134
135
135
136
136
137
137
138
138
139
139
140
140
141
141
142
142
143
143
144
144
145
145
146
146
147
147
148
148
149
149
150
150
151
151
152
152
153
153
154
154
155
155
156
156
157
157
158
158
159
159
160
160
161
161
162
162
163
163
164
164
165
165
166
166
167
167
168
168
169
169
170
170
171
171
172
172
173
173
174
174
175
175
176
176
177
177
178
178
179
179
180
180
181
181
182
182
183
183
184
184
185
185
186
186
187
187
188
188
189
189
190
190
191
191
192
192
193
193
194
194
195
195
196
196
197
197
198
198
199
199
200
200
201
201
202
202
203
203
204
204
205
205
206
206
207
207
208
208
209
209
210
210
211
211
212
212
213
213
214
214
215
215
216
216
217
217
218
218
219
219
220
220
221
221
222
222
223
223
224
224
225
225
226
226
227
227
228
228
229
229
230
230
231
231
232
232
233
233
234
234
235
235
236
236
237
237
238
238
239
239
240
240
241
241
242
242
243
243
244
244
245
245
246
246
247
247
248
248
249
249
250
250
251
251
252
252
253
253
254
254
255
255
256
256
257
257
258
258
259
259
260
260
261
261
262
262
263
263
264
264
265
265
266
266
267
267
268
268
269
269
270
270
271
271
272
272
273
273
274
274
275
275
276
276
277
277
278
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1
```

# Testbench for ARM(2/7)

ISE Project Navigator (O.61xd) - D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Final project\ARM\ARM.xise - [tb\_ARM\_3.v]

File Edit View Project Source Process Tools Window Layout Help

Design View: Implementation Simulation

Hierarchy

- ARM
  - xc6slx25-3ftg256
    - tb\_ARM\_1 (tb\_ARM\_1.v)
    - tb\_ARM\_2 (tb\_ARM\_2.v)
    - tb\_ARM\_3 (tb\_ARM\_3.v)
    - tb\_ARM\_4 (tb\_ARM\_4.v)

```

32   reg clk;
33   reg rst;
34
35   parameter DATA_MEM_SIZE = 64;
36   parameter INS_MEM_SIZE = 32;
37
38   // Instantiate the Unit Under Test (UUT)
39   ARM uut (
40     .clk(clk),
41     .rst(rst)
42 );
43
44
45   integer error, i;
46   reg [31:0]tb_answer[DATA_MEM_SIZE-1: 0];
47
48
49   always #(`CYCLE) clk=~clk;
50
51   initial begin
52     // Initialize Inputs
53     clk = 0;
54     rst = 1;
55     error = 0;
56     #( `CYCLE*2) rst = 0;
57
58     $readmemb(`PROGRAM, uut._ins_mem.mem);
59     $readmemh(`MEM_DATA, uut._data_mem.mem);
60     $readmemh(`ANSWER, tb_answer);
61
62   end
63

```

No Processes Running

Processes: tb\_ARM\_3

- ISim Simulator
  - Behavioral Check Syntax
  - Simulate Behavioral Model

Start Design Files Libraries

Design Summary (out of date) tb\_ARM\_1.v tb\_ARM\_2.v tb\_ARM\_3.v alu.v

Errors

Warnings Errors

Ln 49 Col 18 | Verilog

Check 語法正確

# Testbench for ARM(3/7)

ISE Project Navigator (O.61xd) - D:\Copy\NCKU\NCKU\_SoC\Digital Logic Experiments\Logic Desing 104\Final project\ARM\ARM.xise - [tb\_ARM\_3.v]

File Edit View Project Source Process Tools Window Layout Help

Design View: Implementation Simulation

Hierarchy

- ARM
  - xc6slx25-3ftg256
    - tb\_ARM\_1 (tb\_ARM\_1.v)
    - tb\_ARM\_2 (tb\_ARM\_2.v)
    - tb\_ARM\_3 (tb\_ARM\_3.v) **selected**
    - tb\_ARM\_4 (tb\_ARM\_4.v)

```

32     reg clk;
33     reg rst;
34
35     parameter DATA_MEM_SIZE = 64;
36     parameter INS_MEM_SIZE = 32;
37
38     // Instantiate the Unit Under Test (UUT)
39     ARM uut (
40         .clk(clk),
41         .rst(rst)
42     );
43
44
45     integer error, i;
46     reg [31:0]tb_answer[DATA_MEM_SIZE-1: 0];
47
48
49     always #(`CYCLE) clk=~clk;
50
51     initial begin
52         // Initialize Inputs
53         clk = 0;
54         rst = 1;
55         error = 0;
56         #(`CYCLE*2) rst = 0;
57
58         $readmemb(`PROGRAM, uut._ins_mem.mem);
59         $readmemh(`MEM_DATA, uut._data_mem.mem);
60         $readmemh(`ANSWER, tb_answer);
61
62     end
63

```

No Processes Running

Processes: tb\_ARM\_3

- ISim Simulator
  - behavioral Check Syntax
  - Simulate Behavioral Model** (highlighted)

Start Design Files Libraries

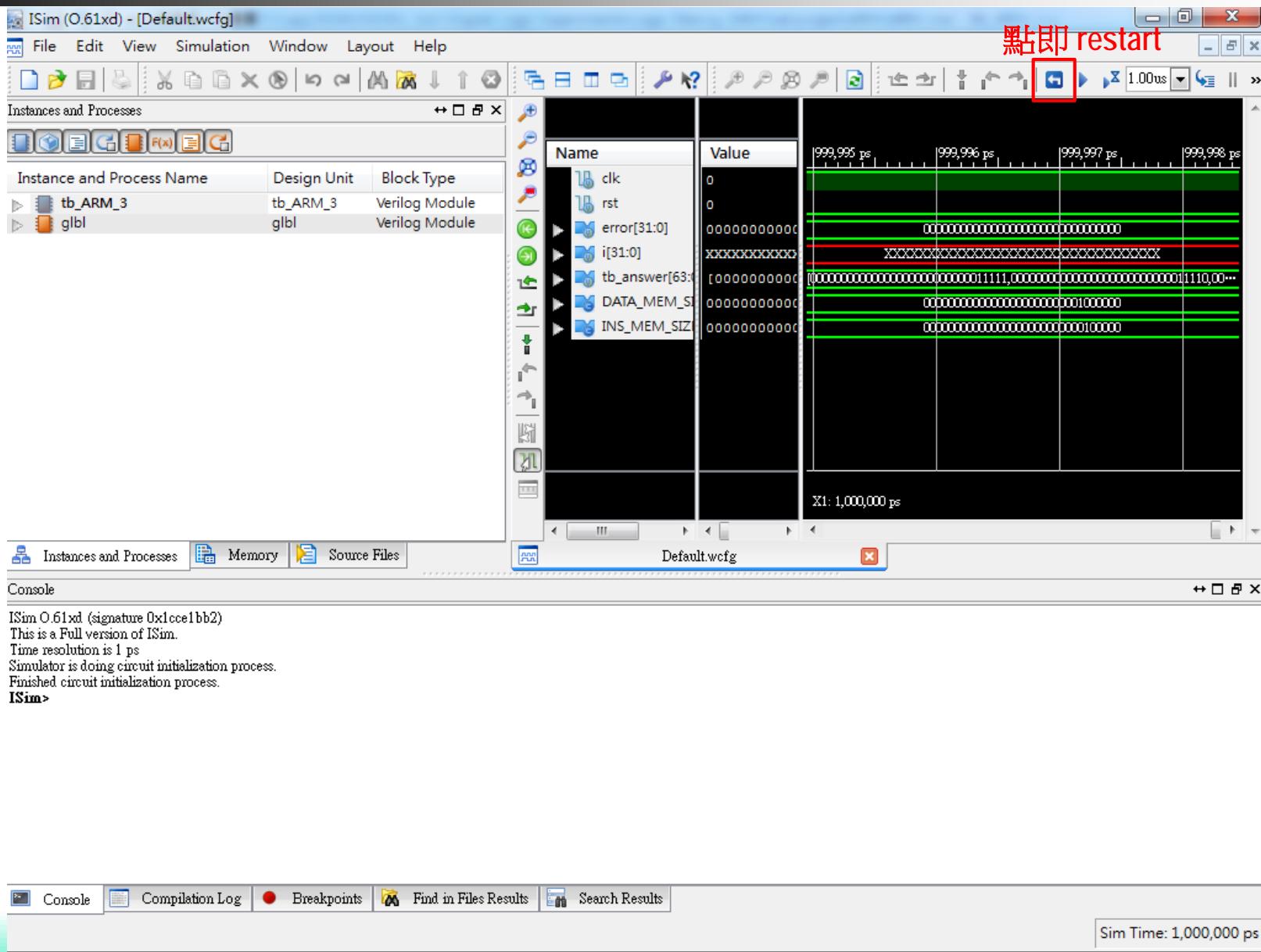
Design Summary (out of date) tb\_ARM\_1.v tb\_ARM\_2.v tb\_ARM\_3.v alu.v

Errors

Warnings Errors

Ln 49 Col 18 Verilog

# Testbench for ARM(4/7)



# Testbench for ARM(5/7)

IISim (O.61xd) - [Default.wcfg]

File Edit View Simulation Window Layout Help

Instances and Processes

Instance and Process Name	Design Unit	Block Type
tb_ARM_3	tb_ARM_3	Verilog Module
glbl	lbl	Verilog Module

點即 run all

Simulator Control Panel

Simulation Waveform View

Name	Value
clk	x
rst	x
error[31:0]	XXXXXXXXXX
i[31:0]	XXXXXXXXXX
tb_answer[63:0]	XXXXXXXXXX
DATA_MEM_SIZ	000000000000
INS_MEM_SIZ	000000000000

Time Scale: 0 ps, 1 ps, 2 ps, 3 ps

X1: 0 ps

Instances and Processes | Memory | Source Files | Default.wcfg

Console

```

IISim O.61xd (signature 0x1cce1bb2)
This is a Full version of IISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
IISim>
# restart
IISim>
  
```

Console | Compilation Log | Breakpoints | Find in Files Results | Search Results | Sim Time: 0 ps

# Testbench for ARM(6/7)

IISim (O.61xd) - [tb\_ARM\_3.v]

File Edit View Simulation Window Layout Help

Instances and Processes

Instance and Process Name	Design Unit	Block Type
tb_ARM_3	tb_ARM_3	Verilog Module
uut	ARM	Verilog Module
Always_49_0	tb_ARM_3	Verilog Process
Initial_51_1	tb_ARM_3	Verilog Process
Always_64_2	tb_ARM_3	Verilog Process
glbl	glbl	Verilog Module

Code Editor (Line 68 to 88):
 

```

68         for(i=0 ;i<DATA_MEM_SIZE; i=i+1)
69             begin
70                 if( uut._data_mem.mem[i] != tb_answer[i] )
71                     begin
72                         error = error + 1'b1;
73                         $display("error at mem[%2d] 0x%h != 0x%h ", i,
74                             end
75
76
77                     if( error == 32'd0)
78                         $display("Congratulation!! All data is correct");
79                     else
80                         $display("You have %d fault !!", error);
81
82                     $finish;
83                 end
84             end
85
86         endmodule
87
88     
```

Source Files tab: tb\_ARM\_3.v

Console output:

```

IISim 0.61xd (signature 0x1cce1bb2)
This is a Full version of IISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
IISim>
# restart
IISim>
# run all
Simulator is doing circuit initialization process.
Finished circuit initialization process.
Congratulation!! All data is correct
全部正確則會顯示此訊息
Stopped at time : 258116 ns : File "D:\Copy\NCKU\NCKU_SoC\Digital Logic Experiments\Logic Desing 104\Final project\ARM\tb_ARM_3.v" Line 82
IISim>
```

Bottom navigation bar: Console, Compilation Log, Breakpoints, Find in Files Results, Search Results

Bottom status bar: Sim Time: 258,116,000 ps | Ln 82 Col 1 | Verilog



# Testbench for ARM(7/7)

ISim (O.61xd) - [tb\_ARM\_3.v]

File Edit View Simulation Window Layout Help

Instances and Processes

Instance and Process Name	Design Unit	Block Type
tb_ARM_3	tb_ARM_3	Verilog Module
uut	ARM	Verilog Module
Always_49_0	tb_ARM_3	Verilog Process
Initial_51_1	tb_ARM_3	Verilog Process
Always_64_2	tb_ARM_3	Verilog Process
glbl	glbl	Verilog Module

Source File Content:

```

68         for(i=0 ;i<DATA_MEM_SIZE; i=i+1)
69             begin
70                 if( uut._data_mem.mem[i] != tb_answer[i] )
71                     begin
72                         error = error + 1'b1;
73                         $display("error at mem[%2d] 0x%h != 0x%h ", i,
74                             end
75
76                     end
77
78             if( error == 32'd0)
79                 $display("Congratulation!! All data is correct");
80             else
81                 $display("You have %2d fault !!", error);
82
83             $finish;
84         end
85
86     endmodule
87
88

```

Console Output:

```

ISim 0.61xd (signature 0x1cce1bb2)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
# restart
ISim>
# run all
Simulator is doing circuit initialization process.
Finished circuit initialization process.
error at mem[32] 0x00000001 != 0x00000000
You have 1 fault !!
Stopped at time : 258116 ns : File "D:/Copy/NCKU/NCKU_SoC/Digital Logic Experiments/Logic Desing 104/Final project/ARM/tb_ARM_3.v" Line 82
ISim> |

```

若有錯誤則會顯示此訊息

Sim Time: 258,116,000 ps | Ln 82 Col 1 | Verilog



# 評分方法

- ◆ 這次期末project占我們總成績的20%，主要分五階段評分!!

階段	分數
繳交程式碼和報告	4%
通過 test_bench_1	4%
通過 test_bench_2	4%
通過 test_bench_3	4%
通過 test_bench_4	4%