(1) RISC and CISC

93 台大電機　94 中央資工

Explain how each of the following six features contributes to the definition of a RISC machine:
(a) Single-cycle operation, (b) Load/Store design, (c) Hardwired control
(d) Relatively few instructions and addressing modes, (e) Fixed instruction format,
(f) More compile-time effort.

(a) Single-cycle operation

所有的動作都限制在一個時脈週期內完成，可以簡化硬體設計並加速指令執行。

(b) Load/Store design

Load/Store 的設計限制CPU 只能以暫存器作為運算元，若運算元是來自記憶體則可使用Load/Store 指令來存取。因為暫存器速度較記憶體快，因此這種方式 可以提高CPU 執行指令的效率。

(c) Hardwired control:相對於微程化的控制單元 Hardwire Control 有較短的指令解碼時間。

(d) Relatively few instructions and addressing modes

可以加快指令及運算元的擷取。

(e) Fixed instruction format

可以加快指令的解碼。

(f) More compile-time effort

RISC 只提供基本指令而無功能較強之指令，因此編譯時會產生較長的程式碼，所以須花較長的編譯時間。

RISC and CISC (96朝陽資工,94淡江資工, 92 長庚資工,91成大電機:RISC)
(94政大資科) (94北科電通) (94元智資工) (93中央資工) (93淡江資工) (92台科電機)
(91東華資工) (90中央電機) (90中央資工)(90中正電機) (90中原資工:RISC)

	RISC	CISC
指令長度	指令長度固定(32 or 64) **(指令長度一致，擷取時速度較快)**	指令的長度長短不一
定址模式	少	多
指令格式	少(指令格式少解碼較容易)	多
存取記憶體	所有運算元都在暫存器執行 (只有load,store 指令能存取記憶體)	大部份指令都能存取暫存器或記憶體的值
控制電路	hardwired control (較簡單)	Microprogrammed control (較複雜)
example	**MIPS**, HP PA-RISC, SPARC Architecture, IBM POWER Architecture, DEC Alpha Family, IBM/Motorola PowerPC, Motorola 88000, ARM, Intel i860, DLX	**Intel 80x86** **Motorola MC68000 Family** DEC PDP-11 DEC VAX IBM 360/370Architecture

(2) little endian

big-endian(MIPS) (94 政大資科) (92 長庚資工)(96 成大電機)

記憶體放置方法,將 **Most** significant byte 放在較低記憶體位址

little-endian (94 政大資科) (92 長庚資工) (96 成大電機)

記憶體放置方法,將 **least** significant byte 放在較低記憶體位址

97 彰師電子 94 清大資工類似

(b)please illustrate big-endian and little-endian by considering the number 0xd75afb68hex stored in memory address 100~103

0xd75afb68hex

address	100	101	102	103
Big endian	d7 hex	5a hex	fb hex	68 hex
Little endian	68 hex	fb hex	5a hex	d7 hex

(3)MIPS code translation

95 交大資工

find binary codes of [add $s4, $t3, $t2] and [lw $s0, 48($t1)].

Reference 1 : add $t0, $s1, $s2 00000010 00110010 01000000 00100000

Reference 2 : lw $t0, 32($s2) 10001110 01001000 00000000 00100000

Instruction	**opcode**	**rs**	**rt**	**rd**	**shamt**	**funct**
R-type	**6**	**5**	**5**	**5**	**5**	**6**
I- type	**6**	**5**	**5**	**16**		

所以 add $s4, $t3, $t2 ➔000000 01011 01010 10100 00000 100000

所以 lw $s0, 48($t1) ➔ 100011 01001 10000 0000000000110000

97 成大資工

Opcode for add,addi,lw be 000000 , 001000 , 100011

Translation the following MIPS code

00000001001010100100000000100000

00100010110101011111111111001110

10001101001010000000010010110000

000000 01001 01010 01000 00000100000 ➔add $s0,$t1,$t2

001000 10110 10101 1111111111001110 ➔addi $s5,$t6,-50

100011 01001 01000 0000010010110000 ➔lw $t0,1200($t1)

p.s. 格式 6,5,5,5,5,6 t0由8開始 s0由16開始

opcode add:0 sub:0 lw:35 sw:43 beq:4 bne:5 j:2

j 10000 ➔000010 00000000000000100111000100 (後面為 2500的binary)

j 數值翻成code除4,從code翻出來乘4

(4)addressing mode 96 暨南資工定義 93 交大資工畫 94 清大電機,95 交大資工(那類指令)

(a)explain (b)draw the five MIPS addressing mode

(1) **Immediate addressing**, (addi,andi,slti,ori) (make the common case fast)

運算元(operand)爲instruction中的constant

(2) **Register addressing** (add,sub,jr,and,or,slt) (simplicity favors regularity)

運算元(operand) 在 register 中

(2)**Base or displacement addressing** (lw,sw,lb,sb) (make the common case fast)

運算元(operand)在記憶體中

記憶體位址爲register與instruction中的constant之和

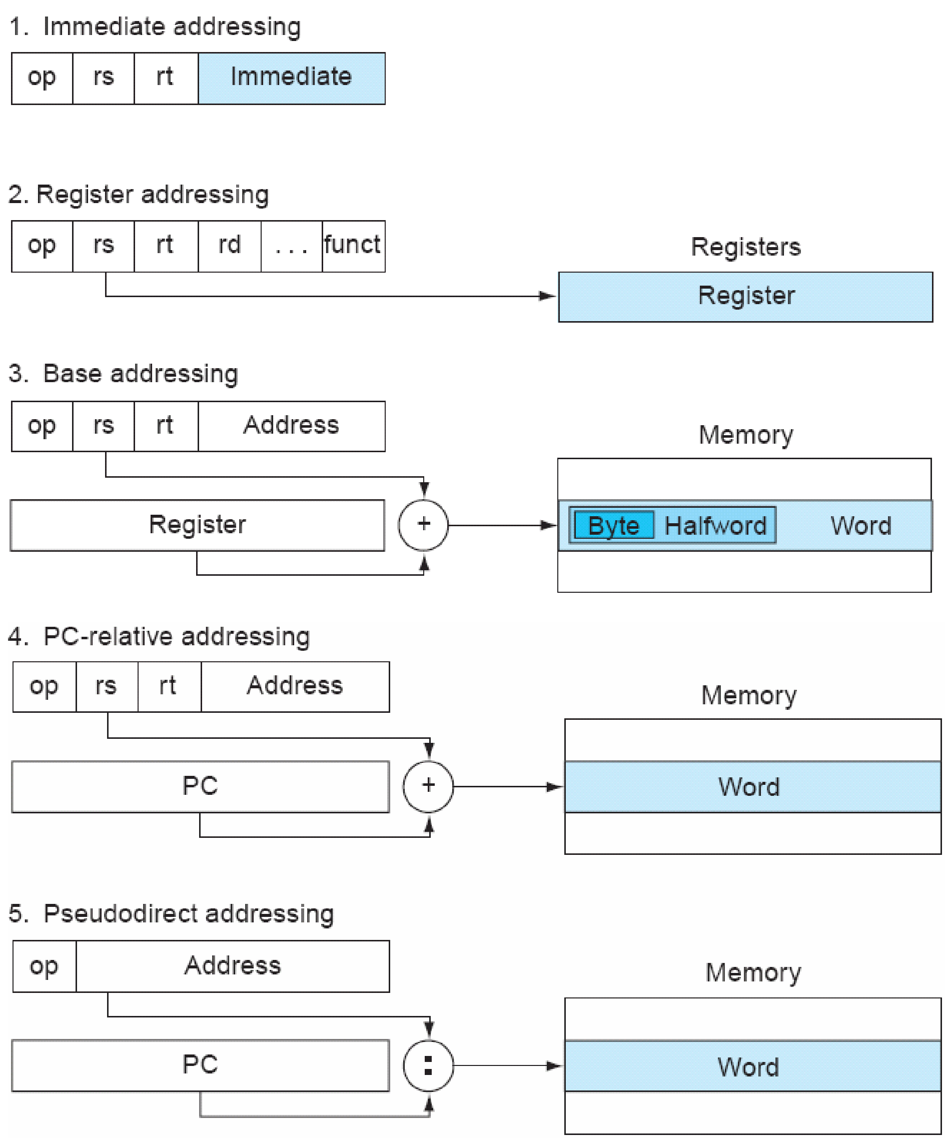
(4) **PC-relative addressing** (beq,bne)

運算元(operand)在記憶體中

記憶體位址爲PC與instruction中的constant之和

(5) **Pseudodirect addressing** (j,jal)

跳躍位址爲 instruction 中的 26 bit 與 PC 的 高位元結合



(5) number and overflow

n bit 2`s complement range： $-2^{n-1} \sim 2^{n}-1$

96 台大資工

Number representation. What range of integer number can be represented by 16-bit 2's complement number?
$2^{-15} \sim 2^{15}-1$

94 暨南資工

Rewrite -100$_{ten}$ using a 16-bit binary representation use (1) sign and magnitude representation (2) one`s complement representation (3) two`s complement representation
sign and magnitude representation：1000 0000 0110 0100 one`s complement representation: 1111 1111 1001 1011 two`s complement representation: 1111 1111 1001 1100

93 清大電機

Explain why most of today`s computer system use 2`s complement instead of signed-magnitude in their hardware implementation
(1)sign bit 要放在最左邊或最右邊如何決定 (2)加法器需要額外一個步驟判斷正負 (3)有正 0 與負 0 之分,粗心的程式設計師容易犯錯

overflow conditions (96 清大電機) (91 政大資科)

Operation	A	B	Result
A+B	≥0	≥0	＜0
A+B	＜0	＜0	≥0
A-B	≥0	＜0	＜0
A-B	＜0	≥0	≥0

95 清大電機

How nops instruction implement in MIPS machine?
sll $0,$0,0

(6) floating point number representation

92 台師資工

Explain the IEEE 754 floating point number standard

Single precision: sign bit:1 exponent bit:8 significand bit:23 bias: 127

表示法: $(-1)^s * (1 + significand) * 2^{E-127}$

double precision: sign bit:1 exponent bit:11 significand bit:52 bias: 1023

表示法: $(-1)^s * (1 + significand) * 2^{E-1023}$

92 政大資科

Why is biased notation used in IEEE 754 representation?

多數浮點數運算需要先比較指數大小來決定要對齊哪一個數值的指數，使用Biased表示法可以直接比較其數值即可判斷兩個浮點數的大小而不須考慮正負號，因此可以加快比較速度

(96台大電機) (96大同資工) (95元智資工類似) (95北科電通) (92雲科電子類似)
(91淡江資工類似) (91淡江電機類似) (90中正電機類似) (90中原資工類似)

Show the single-precision IEEE 754 binary representation of the number -0.75 in single and double precision.

single precision.

-0.75_{10}=$-0.11_2 = -1.1 * 2^{-1}$

$(-1)^s * (1 + significand) * 2^{exponent-127}$

$\Rightarrow s = 1, exponent = -1 + 127 = 126$

double

-0.75_{10}=$-0.11_2 = -1.1 * 2^{-1}$

$(-1)^s * (1 + significand) * 2^{exponent-1023}$

$\Rightarrow s = 1, exponent = -1 + 1023 = 1022$

s	exponent								significand
1	0	1	1	1	1	1	1	0	10……. 0

s	exponent											significand
1	0	1	1	1	1	1	1	1	1	1	0	10….0

(92雲科電子類似)

What decimal number is represented by this word?

31	30	29	28	27	26	25	24	23	22….. 0
1	1	0	0	0	0	0	0	1	01….. 0

$(-1)^s * (1 + significand) * 2^{exponent\text{-}127}$

$= (-1)^1 * (1 + 0.010...) * 2^{128\text{-}127}$

$= -1.01 * 2^2 = -101_2 = -5_{10}$

94 台大電機

(a) what is denormalized number ? (b) show how to use gradual underflow to represent a denorm in a floating point number
(a)當發生數値發生 exponent underflow 無法被 normalized form 表示時,用 denormalized number 來表示數值, 在 IEEE754 中,exponent 最小=0 ,但 significand 為非 0 (b) 允許一個有效數字在精確度上逐漸遞減，一直到變成 0 爲止稱 gradual underflow 如 smallest normalized number: ± 1.000 0000 0000 0000 0000 0000 × 2^{-126} 但 smallest denormalized number: ± 0.000 0000 0000 0000 0000 0001 × 2^{-126}=2^{-149}

IEEE 754 encoding of floating-point numbers (95成大資工)

Object represented	**Single precision**		**Double precision**	
	Exponent	Signficand	Exponent	Signficand
0	0	0	0	0
± denormalized number	0	nonzero	0	nonzero
± floating-point number	1~254	anything	1~2046	anything
± infinity	255	0	2047	0
NaN (Not a Number)	255	nonzero	2047	nonzero

IEEE single precision (96 交大資聯) (93 政大資科)

smallest normalized number	± 1.000 0000 0000 0000 0000 0000 × 2^{-126}
largest normalized number	± 1.111 1111 1111 1111 1111 1111 × 2^{127}
smallest denormalized number	± 0.000 0000 0000 0000 0000 0001 × 2^{-126} (2^{-149})
largest denormalized number	± 0. 111 1111 1111 1111 1111 1111 × 2^{-126}

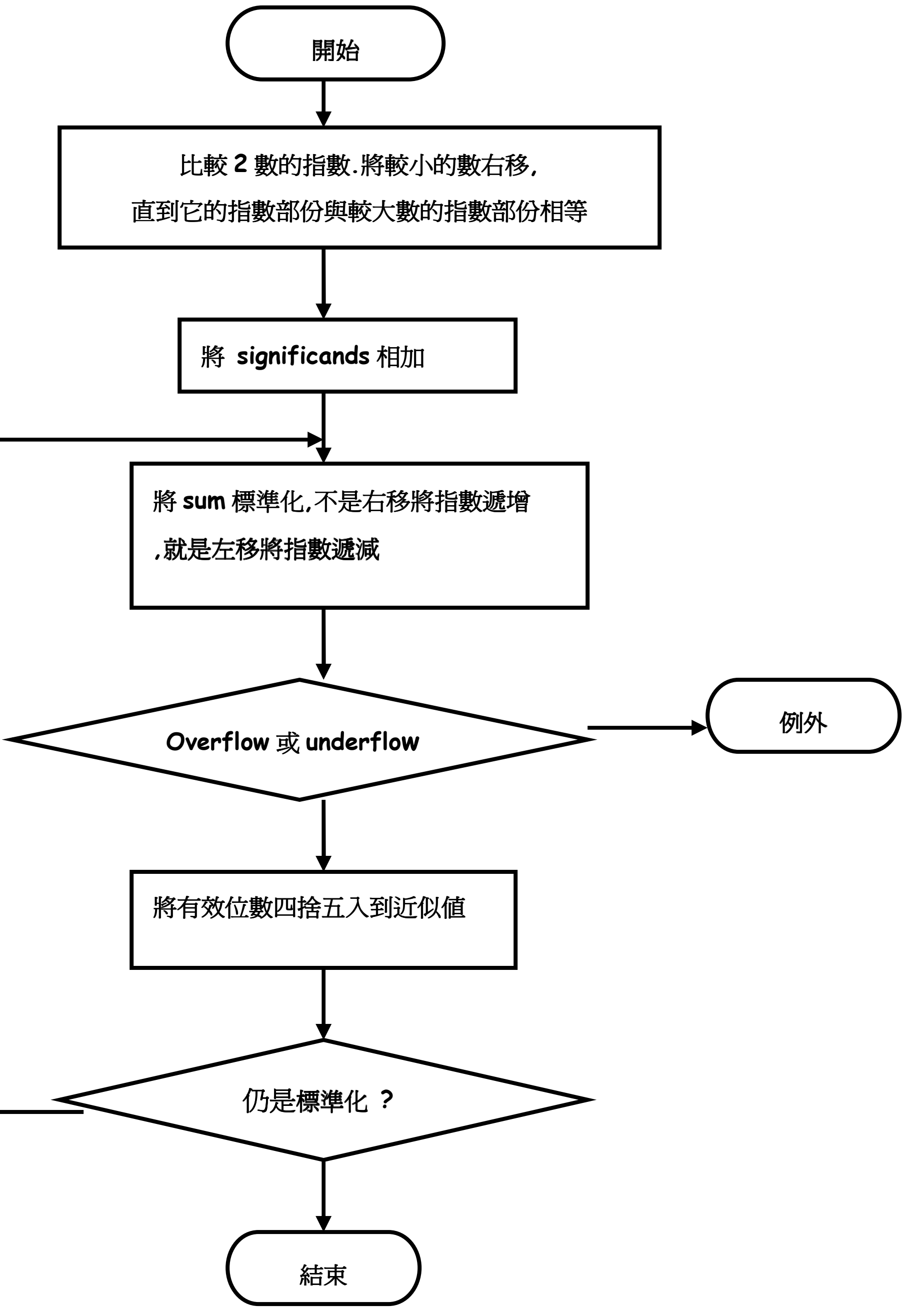
	31	30	29	28	27	26	25	24	23	22.......0
smallest normalized number	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	00.......0
largeest normalized number	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	11.........1
smallest denormalized number	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	00........1
largest denormalized number	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	11..........1

96 交大資工

Suppose there was a 16-bit IEEE 754-like floating-point format with 5 exponent bits (±1.0000 0000 00 × 2^{-15} to ±1.1111 1111 11 × 2^{14}, ±0, ±∞, NaN) is the likely range of numbers it could represent.
False :用 32-bit IEEE 754 floating-point去想 $-126 \le exp \le 127$ n bit exponent : $-(\frac{2^n}{2}-2) \le exp \le (\frac{2^n}{2}-1)$ (exponent 頭尾留給特殊表示法) $5\ \ bit: -14 \le exp \le 15$

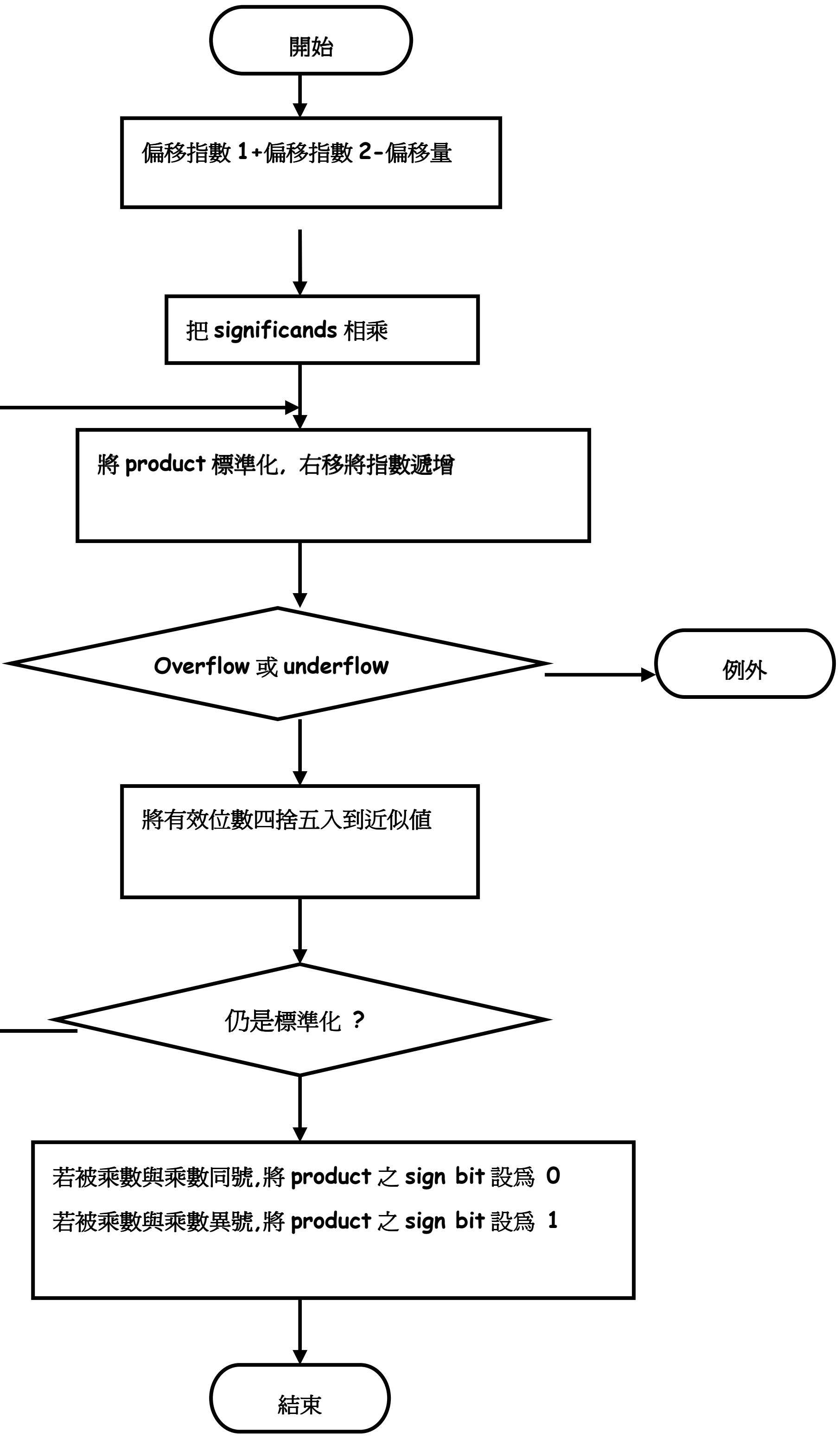
(7) Floating-Point addition operation (94 彰師電子) (93 暨南資工,類似) (90 暨大資工)

Floating-Point Addition



開始
比較 2 數的指數.將較小的數右移,
直到它的指數部份與較大數的指數部份相等
將 significands 相加
將 sum 標準化,不是右移將指數遞增
,就是左移將指數遞減
Overflow 或 underflow
例外
將有效位數四捨五入到近似值
仍是標準化 ?
結束

Floating-Point Multiplication


開始
偏移指數 1+偏移指數 2-偏移量
把 significands 相乘
將 product 標準化，右移將指數遞增
Overflow 或 underflow
例外
將有效位數四捨五入到近似值
仍是標準化 ?
若被乘數與乘數同號,將 product 之 sign bit 設爲 0
若被乘數與乘數異號,將 product 之 sign bit 設爲 1
結束

(95元智資工 2.56 + 2.34)

Example

(1) add 0.5 and -0.4375

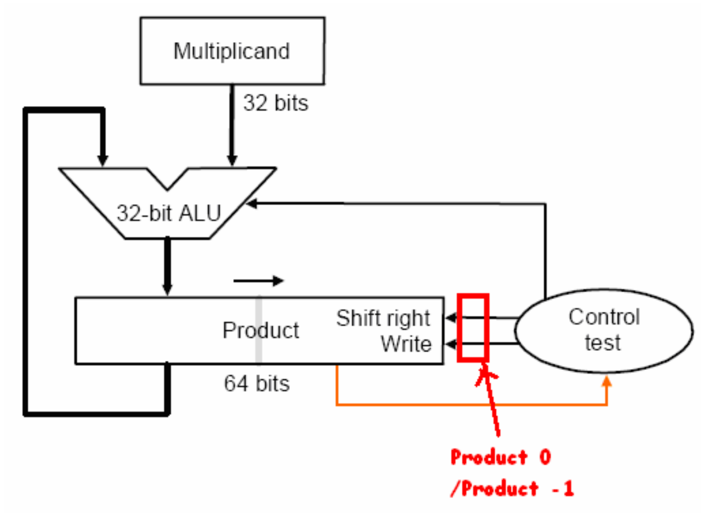
(2) multiply 0.5 and -0.4375 (92 政大資科)

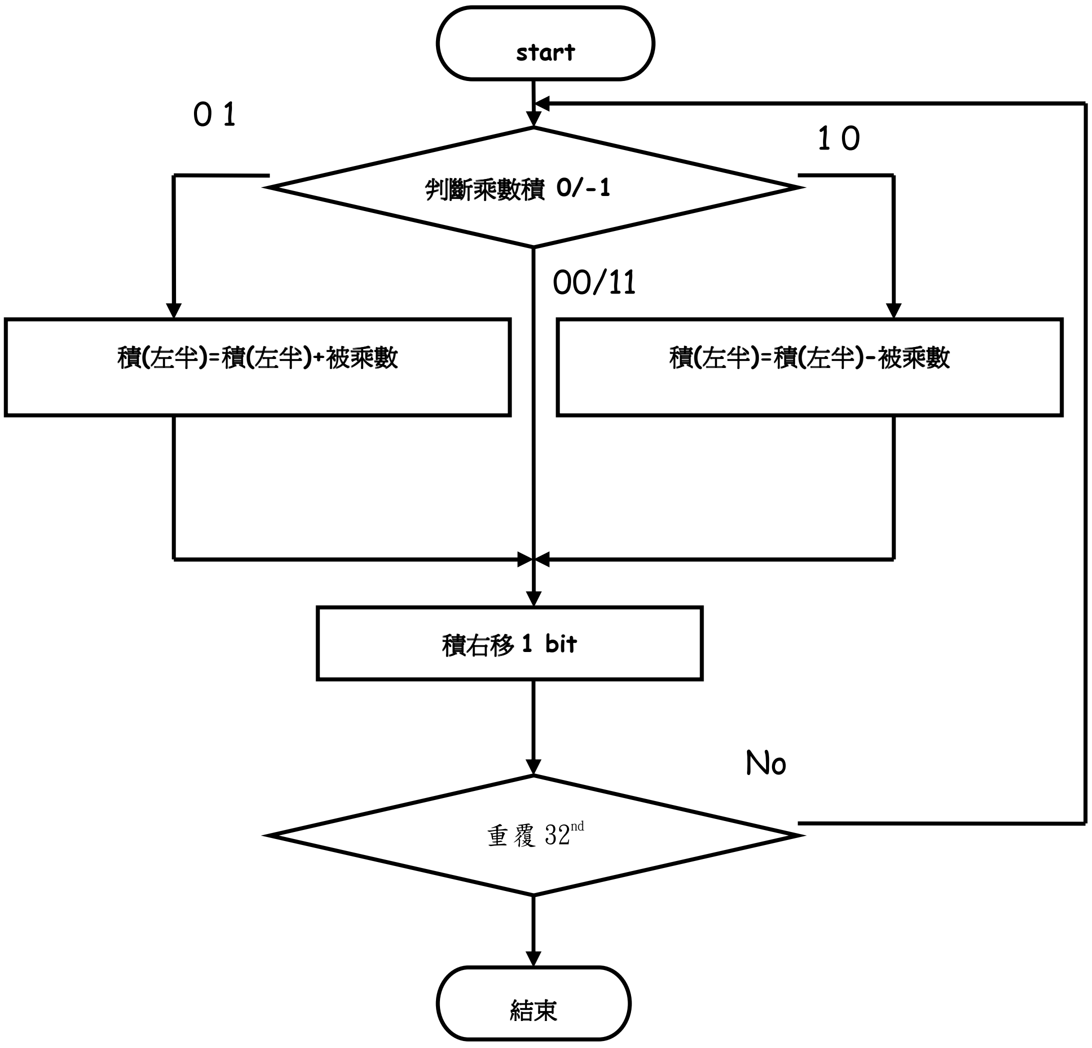
step	
0	$0.5 = 1.000 \times 2^{-1}$, $-0.4375 = -1.110 \times 2^{-2}$
1	將較小的數右移,直到它的指數部份與較大數的指數部份相等 ➔ $-0.4375 = 1.110 \times 2^{-2} = -0.111 \times 2^{-1}$
2	將 **significands** 相加 ➔ $1.000 \times 2^{-1} + 0.111 \times 2^{-1} = 0.001 \times 2^{-1}$
3	將 **sum** 標準化,偵測 **overflow or underflow** ➔ $0.001 \times 2^{-1} = 1.000 \times 2^{-4}, -126 \le -4 \le 127$,so no overflow or underflow
4	將有效位數四捨五入到近似值 ➔ $0.001 \times 2^{-1} = 1.000 \times 2^{-4}$
5	$1.000 \times 2^{-4} = 0.0625_{10}$

step	
0	$0.5 = 1.000 \times 2^{-1}$, $-0.4375 = -1.110 \times 2^{-2}$
1	偏移指數 1+偏移指數 2-偏移量 ➔ $(-1+127)+(-2+127)-127 = -3+127 = 124$
2	把 **significands** 相乘 ➔ $1.000 \times 1.110 = 1.110 \times 2^{-3}$
3	將 **product** 標準化,偵測 **overflow or underflow** ➔ $-126 \le -3 \le 127$,so no overflow or underflow
4	將有效位數四捨五入到近似值 ➔ 1.110×2^{-3}
5	若被乘數與乘數異號,將 **product** 之 **sign bit** 設爲 **1** $-1.110 \times 2^{-3} = -0.21875_{10}$

(8) booth algorithm

Booth algorithm





Booth's Algorithm (94,92 交大資工) (92 交大資工) (94 成大資工,91 政大資科:prove)

Prove of Booth's Algorithm

(a_{i+1}, a_i) 00→nop 01→sub multiplicand 10→add multiplicand 11→nop

The sum of booth is

$(a_{-1} - a_0) \times b +$

$(a_0 - a_1) \times b \times 2^1 +$

$(a_1 - a_2) \times b \times 2^2 +$

……………

$(a_{29} - a_{30}) \times b \times 2^{30} +$

$(a_{30} - a_{31}) \times b \times 2^{31}$

因爲 $-a_i \bullet 2^{i-1} + a_i \bullet 2^i = 2^{i-1}(a_i \bullet 2 - a_i) = a_i 2^{i-1}$

所以上式化簡爲 $b \times [a_0 \bullet 2^0 + a_1 \bullet 2^1 + ... + a_{30} \bullet 2^{30} - a_{31} \bullet 2^{31}] = b \times a$ (括號內爲 a 的 2`s)

when booth`s algorithm has better performance? Worse performance?

better performance? a sequence 0 or 1 could lower the addition/subtraction

worse performance? 0 and 1 appear alternative

Example: 2*-3 booth algorithm(93 清大通訊) (94 中山資工)

(93 台大電機 2*-6) (94 淡江電機)(90 中正電機-3*-7)

iteration	step	被乘數	積
0	Initial values 被乘數放入積	0010	0000 1101 **0**
1	10: no operation	0010	**1110** 1101 0
	shift right product	0010	**1111 0110 1**
2	01: prod= prod - mcand	0010	**0001** 0110 1
	shift right product	0010	**0000 1011 0**
3	10 : no operation	0010	**1110** 1011 0
	shift right product	0010	**1111 0101 1**
4	11 : prod= prod + mcand	0010	1111 0101 1
	shift right product	0010	**1111 1010 1**

96 交大資工,95 交大資工,92 交大資工 92,93 中央資工

Please explain the concept of non-restoring division algorithm

Restoring的除法演算法中, 若被除數減完除數的結果為負，則我們必須做restore的動作➔

Step 1:除數加回以恢復被除數的值$(r + d)$

Step 2:接著將被除數向左移一位元$((r + d) \times 2)$

Step 3:並於下一回合減掉除數$[(r + d) \times 2)-d]$

Nonrestoring的除法演算法中，當被除數減完除數的結果若為負時，我們並不馬上做restore的動作(將除數加回被除數)

Step 1:將被除數向左移一位元$(r \times 2)$

Step 2:於下一回合加上除數$(r \times 2 + d)$

non-restoring performance的原因爲 :因為$(r + d) \times 2 - d = r \times 2 + d$,假設加法和減法所花的時間是一樣的，則使用Nonrestoring的方式可以在一個回合的計算中少做一次減法，因而其performance較佳。

97 交大資工

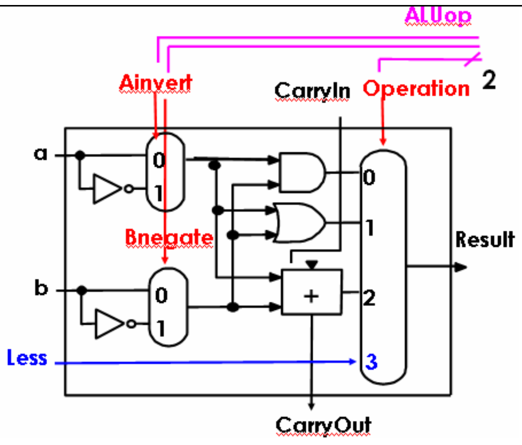
Using a 4-bit version of the algorithm to save pages,let`s try dividing 7_{10} by 2_{10}, or $0000\ 0111_2$ by 0010_2 (first divide algo)

iter	**Step**	商數 (Q)	除數(D)	餘數(R)
0	Initial values 餘數擺被除數	0000	0010 0000	0000 0111
1	1:餘左半🡐餘左半−除左半	0000	0010 0000	**1**110 0111
	2b:餘<0 ⇒ 餘左🡐餘左+除左,商左移,lsb=0	000**0**	0010 0000	0000 0111
	3 ：除數右移	0000	**0001 0000**	0000 0111
2	1: 餘左半🡐餘左半−除左半	0000	0001 0000	**1**111 0111
	2b： 餘<0 ⇒ 餘左🡐餘左+除左,商左移,lsb=0	000**0**	0001 0000	0000 0111
	3 ：除數右移	0000	**0000 1000**	0000 0111
3	1: 餘左半🡐餘左半−除左半	0000	0001 0000	**1**111 1111
	2b： 餘<0 ⇒ 餘左🡐餘左+除左,商左移,lsb=0	000**0**	0001 0000	0000 0111
	3 ：除數右移	0000	**0000 0100**	0000 0111
4	1: 餘左半🡐餘左半−除左半	0000	0000 0100	**0**000 0011
	2a： 餘≥ 0 ,商左移,**lsb=1**	000**1**	0000 0100	0000 0011
	3 ：除數右移	0001	**0000 0010**	0000 0011
5	1: 餘左半🡐餘左半−除左半	0001	0000 0010	**0**000 0011
	2a： 餘≥ 0 ,商左移,**lsb=1**	001**1**	0000 0010	0000 0001
	3 ：除數右移	0011	**0000 0001**	0000 0001

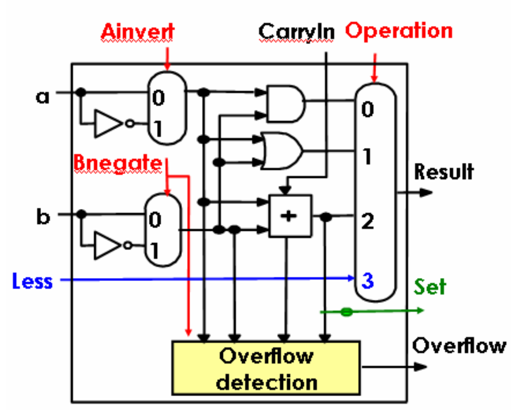
(9)ALU

91 暨南資工

Design a 8-bit ALU that can perform AND, OR, XOR, addition, and subtraction.

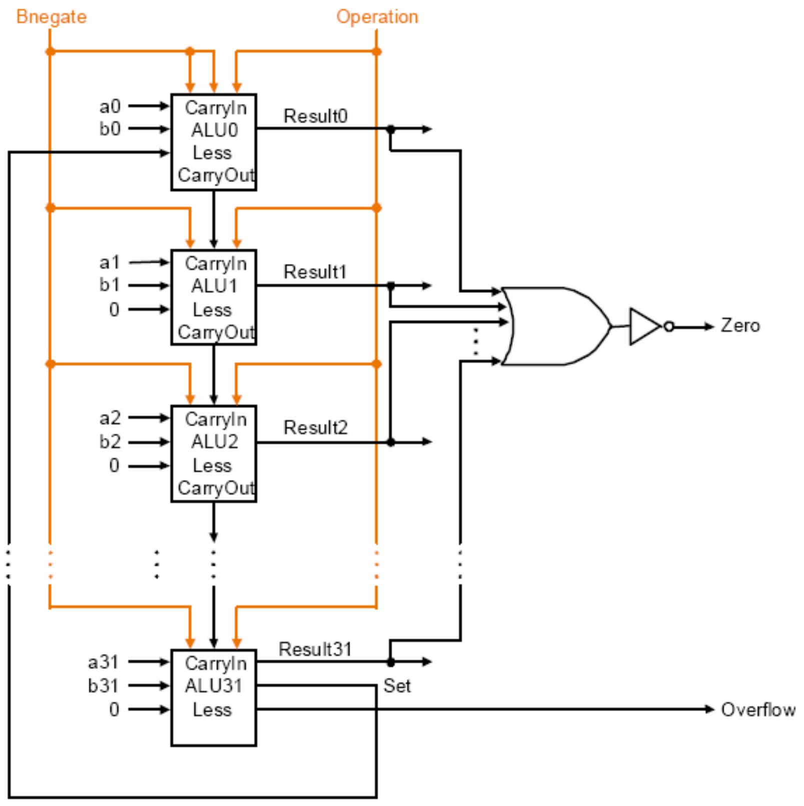


1-bit ALU: and,or,add,a⊕b,slt



1-bit ALU: and,or,add,a⊕b,slt

Overflow detect



(10) three factors impact on performance

96 交大資工,96 中正資工,96 高應電機,95 清大電機

The law of performance indicates that CPU time can be shown as a product of 3 terms. What are those three terms? Explain what factors may have impact on the three terms respectively

performance 三要素:Instruction Count ,CPI,Clock Cycle Time

$$CPU\ \ Clock\ \ Cycle = \sum_{i=1}^{n} CPI_i \times IC_i$$

	Instr. Count	CPI	Clock Rate
Algorithm	V	(V)	
Programing Language	V	V	
compiler	V	V	
Instr. Set architecture	V	V	V
Organization		V	V
Technology			V

(11)performance 計算題　　2 版習題

Consider two different implementations, M1 and M2, of the same instruction set. There are four classes of instruction (A, B, C, and D) in the instruction set. M1 has a clock rate of 500 MHz and M2 has a clock rate of 750 MHz.

Instruction class	CPI (MachineM1)	CPI (Machine M2)
A	**1**	**2**
B	**2**	**2**
C	**3**	**4**
d	**4**	**4**

(1) Assume the peak performance is defined as the fastest rate that a machine can execute an instruction sequence chosen to maximum that rate. What are the peak performances of M1 and M2? Please express as instructions per second?

(2) If the number of instructions executed in a certain program is divided equally among the classes of instructions. How much faster is M2 than M1?

(1) Peak performance for M1 = $(500 \times 10^6)/1 = 500 \times 10^6$

Peak performance for M2 = $(750 \times 10^6)/2 = 375 \times 10^6$

(2)CPI for M1 = (1 + 2 + 3 + 4)/4 = 2.5

CPI for M2 = (2 + 2 + 4 + 4)/4 = 3

Suppose the instruction count the program = IC

Execution for M1 = (2.5 × IC)/500 × 106 = 5 × IC (ns)

Execution for M2 = (3 × IC)/750 × 106 = 4 × IC (ns)

M2 is faster than M1 by 5/4 = 1.25 times

96 中山資工 , 91 東華資工,90 逢甲資工算盤習題

A compiler designer is trying to decide between two code sequences for a particular machine. The hardware designers have supplied the following facts:

Instruction class	CPI for this instruction class
A	1
B	2
C	3

For a particular high-level-language statement, the compiler writer is considering two code sequences that require the following instruction counts:

Code sequence	Instruction counts for instructuion class		
	A	B	C
1	2	1	2
2	4	1	1

Which code sequence executes the most instructions? Which will be faster?
What is the CPI for each sequence?

Instruction count for code sequence 1=2+1+2=5
Instruction count for code sequence 2=4+1+1=6

$$CPU\ \ clock\ \ cycle_A = 1\times2\ +\ 2\times1\ +\ 3\times2 = 10$$

$$CPU\ \ clock\ \ cycle_B = 1\times4\ +\ 2\times1\ +\ 3\times1 = 9$$

$$\frac{Performance_B}{Performance_A} = \frac{Execution\ \ Time_A}{Execution\ \ Time_B} = \frac{10}{9} = 1.1$$

$$\Rightarrow B\ \ is\ \ 1.1\ \ times\ \ faster\ \ than\ \ A$$

$$CPI_A = \frac{1\times2\ +\ 2\times1\ +\ 3\times2}{2+1+2} = 2$$

$$CPI_B = \frac{1\times2\ +\ 2\times1\ +\ 3\times2}{4+1+1} = 1.5$$

$$Average\ \ CPI = \frac{\sum CPI * Instruction\ \ Count}{\sum Instruction\ \ Count}$$

96 中山資工 97 中央電機　算盤習題

Consider the machine with three instruction classes and CPI measurements from the last example on page 64. Now suppose we measure the code for the same program from two different compilers and obtain the following data:

Instruction class	CPI for this instruction class
A	1
B	2
C	3

Code from	Instruction counts (in billions) For each instruction class		
	A	B	C
Compiler 1	5	1	1
Compiler 2	10	1	1

Assume that the machine's clock rate is 500 MHZ. Which code sequence will execute Faster according to MIPS? According to execution time?

$$CPU\ \ Clock\ \ cycle_1 = (5*1+1*2+1*3)*10^9 = 10*10^9$$

$$CPU\ \ Clock\ \ cycle_2 = (10*1+1*2+1*3)*10^9 = 15*10^9$$

$$Execution\ \ Time_1 = \frac{CPU\ \ Clock\ \ cycle_1}{CPU\ \ Clock\ \ rate} = \frac{10*10^9}{500*10^6} = 20$$

$$Execution\ \ Time_2 = \frac{CPU\ \ Clock\ \ cycle_2}{CPU\ \ Clock\ \ rate} = \frac{15*10^9}{500*10^6} = 30$$

$$CPI_1 = \frac{(5*1+1*2+1*3)*10^9}{(5+1+1)*10^9} = \frac{10}{7} \qquad CPI_2 = \frac{(10*1+1*2+1*3)*10^9}{(10+1+1)*10^9} = \frac{5}{6}$$

$$MIPS_1 = \frac{Instruction\ \ Count_1}{Execution\ \ Time_1*10^6} = \frac{(5+1+1)*10^9}{20*10^6} = 350$$

$$MIPS_1 = \frac{Instruction\ \ Count_2}{Execution\ \ Time_2*10^6} = \frac{(10+1+1)*10^9}{30*10^6} = 400$$

We are interested in two implementations of a machine one with and one without special floating-point hardware. Consider a program, P, with the following mix of operations:

floating-point multiply	10%
floating-point add	15%
floating-point divide	5%
integer instructions	70%

Machine MFP (Machine with Floating Point) has floating-point hardware and can therefore implement the floating-point operations directly. It requires the following number of clock cycles for each instruction class:

floating-point multiply	6
floating-point add	4
floating-point divide	20
integer instructions	2

Machine MNFP (Machine with No Floating Point) has no floating-point hardware and so must emulate the floating-point operations using integer instructions. The integer instructions all take 2 clock cycles. The number of integer instructions needed to implement each of the floating-point operations is as follows:

floating-point multiply	30
floating-point add	20
floating-point divide	50

Both machines have a clock rate of 1000 MHz. Find the native MIPS ratings for both machines.

CPI for MFP=0.1*6+0.15*4+0.05*20+0.7*2=3.6

CPI for MNFP be 2 (題目給的)

MIPS for MFP=1000/3.6=278　　MIPS for MNFP =1000/2=500

(12)MIPS

$$MIPS = \frac{Instruction\ \ Count}{Execution\ \ Time * 10^6} = \frac{clock\ \ rate}{CPI * 10^6}$$

(96 交大資工) (94 清大資工) (94 中山資工) (93 清大電機) (93 中央通訊) (93 清大電機)

Pitfall:using MIPS as a performance metric
(1)MIPS 只描述指令速度,並未描述只令本身的能力 (2)再同一台電腦,不同的程式會有不同的 MIPS (3)MIPS能與performance成反比

(95 中央電機)

A performance metric on processor is called "MOPS". What is MOPS? If a machine has the same metric on MIPS and MOPS, what does it means?
(a) MOPS: Millions operations per second (b) 代表一個指令只有一個operation,即為single cycle machine

(13) Amdahl`s Law (96 朝陽資工) (96 彰師資工) (96 北科電通) (96 暨南資工) (95 朝陽資工) (92 大同資工) (90 中央資工)

(a)Definition：一個系統中總效能的改善,會受限於改善部份所佔有系統的比例

(b)*Amdahl`s law formula*

$$(1) speed\ \ up = \frac{效能_{改良後}}{效能_{改良前}} = \frac{執行時間_{改良前}}{執行時間_{改良後}} = \frac{1}{\frac{改良\%}{倍率} + 未改良\%}$$

$$(2) 執行時間(Execution\ \ time)_{改良後} = \frac{改良的執行時間}{改良部分佔系統比例} + 未改良的執行時間$$

(c)Amdahl law ➔ Design principle 中的 make the common case fast

(14) add and Memory for datapath 96 交大資工 95 交大資工

To implement these five MIPS instructions: [lw, sb, addi, xor, beq],

(a) If simple single-cycle design is used, at least how many adders must be used?
What each of these adders is used for?

(b)Similarly, at least how many memories are there? What each of them is used for?

(c) Repeat (a) for multi-cycle design.

(d) Repeat (b) also for multi-cycle design.

(a)2個adder ,一個用於PC+4,另一個用於 branch target address

(b) 2個memory,一個data memory, 一個 instruction memory

(c) 1個ALU(adder),有 PC+4, branch target address,算術運算的功能

(d) 1 個 memory 用於 data memory 也用於 instruction memory

(95 中山資工) (92 清大資工)(90 暨大資工)

MIPS instructions classically take five steps to execute

(1) IF :instruction fetch from memory
(2) ID : instruction decode and register file Read
(3) Ex : Execution or calculate the address
(4) Mem :data memory access
(5) Wb : Write Back to the register file

(15)MIPS instruction 94 清大電機 , 97 交大資工

Step name	Action for R-type instructions	Action for memory-reference instructions	Action for branches	Action for jumps
Instruction fetch	IR = Memory[PC]			
	PC = PC + 4			
Instruction decode/register fetch	A = Reg [IR[25-21]]			
	B = Reg [IR[20-16]]			
	ALUOut = PC + (sign-extend (IR[15-0]) << 2)			
Execution, address computation, branch/ jump completion	ALUOut = A op B	ALUOut = A + sign-extend (IR[15-0])	if (A ==B) then PC = ALUOut	PC = PC [31-28] II (IR[25-0]<<2)
Memory access or R-type completion	Reg [IR[15-11]] = ALUOut	Load: MDR = Memory[ALUOut] or Store: Memory [ALUOut] = B		
Memory read completion		Load: Reg[IR[20-16]] = MDR		

(16)control line signal

(96台大電機) (94交大資工) (92元智資工)

	ALUOp1	ALUOp0	ALUSrc	Branch	MemRead	MemWrite	MemtoRe	RegWrite	RegDst
R-type	1	0	0	0	0	0	0	1	1
lw	0	0	1	0	1	0	1	1	0
sw	0	0	1	0	0	1	X	0	X
beq	0	1	0	1	0	0	X	0	X
	8	1	6	1	4	2	4	C	8

(17)code and CPI

93 中山資工

```
            add   $t0,$zero,$zero
loop1:      add   $t1,$t0,$t0
            add   $t1,$t1,$t1
            add   $t2,$a0,$t1
            sw    $zero,0($t2)
            addi  $t0,$t0,1
            slt   $t3,$t0,$a1
            bne   $t3,$zero,loop1
```

average CPI,a1=10 initial?

$$\frac{4+(4+4+4+4+4+4+3)*10}{1+7*10}=3.86$$

97 交大資工 算盤習題

The mix is 25% loads , 13% stores , 19% branchs, 2% jump ,%43 ALU

Clock cycle for each : loads=5, stores=4, ALU=4, branchs=3,jump=3

What the variable-clock cycle CPU ? and fixed-clock cycle CPU ?

CPU clock cycle = 5 × 25% + 4×13%+ 4× 43%+ 3× 19% + 3× 2% = 4.12 ns

fixed-clock cycle CPU➔5ns

(96 成大資工)

Suppose we have a floating-point unit that requires 8ns for floating-point unit and 16ns for a floating-point multiply.All the other functional unit times are as in the previous example,and a floating-point instruction is like an arithmetic-logical instruction,except that it uses the floating-point ALU rather than the main ALU.Find the performance ration between an implementation in which the clock cycle is different for each instruction class and an implementation in which all instruction have the same clock cycle time ,Register :1 ns,Mwmory:2ns ,ALU:2ns

All loads take the same time and comprise 31% of the instruction

All stores take the same time and comprise 21% of the instruction

R-format instruction comprise 27% of the mix

Branchs comprise 5% of the instruction,while jumps comprise 2%

FP add and subtract take the same time and together total 7% of the instruction

FP multiply and divide take the same time and together total 7% of the instruction

8 × 31% +7 × 21% + 6 × 27% + 5 × 5% + 2 × 2% + 20 × 7% + 12 × 7% = 8.1 ns.

	Memory	Register	ALU , FP add/multiple	Memory	Register	Delay(ns)
load	2	1	2	2	1	8
store	2	1	2	2	X	7
R-format	2	1	2	X	1	6
Branch	2	1	2	X	X	5
Jump	2	X	X	X	X	2
FP add/sub	2	1	8	X	1	12
FP mul/div	2	1	16	X	1	20

$$\frac{Performance_{variable}}{Performance_{fixed}} = \frac{20}{8.1} = 2.47$$

(18)Compare Micro-programming control and Hardwired control (94 中山資工)

(95 中原資工 explain both)

	Hardwired	**Micro-programming**
	又稱有限狀態機法(Finite State Machine: FSM)。 利用 FSM 將指令執行之過程分成數個狀態，並分析每個狀態要產生的控制訊號，決定下一個時脈週期要轉換到哪一個狀態。最後用硬體來製作控制電路	利用微指令(microinstruction)，來定義每一個狀態的控制訊號，執行一個微指令就會送出此指令所描述的控制訊號
prons	速度快 硬體成本便宜	修改設計容易 能處理較複雜的指令集
cons	修改設計不容易 只適合處理較簡單的指令集	速度慢 只適合處理較簡單的指令集 硬體成本較高

(19)pipeline

97 中央電機

Pipeline advantage and disadvantage
Advantage:透過 instruction overlap execution 增加 throughput Disadvantage:硬體電路較複雜,成本較高,需解決 hazard

92 清大資工,95 交大電子,96 交大電子,90 中央資工

(1)what is the ideal performance improment for an n-stage pipeline machine? (2) Why pipeline not archieve ideal ?
(1)n times faster than the machine without pipeline (2)每個 stage 爲必能切割到完全相等 Pipeline 有 overhead (如 pipeline register 的時間) 在開始時,把資料填進 pipeline 需要時間 Pipeline 有 hazard (structural,data,control)

(20)pipeline hazard

(94 中正資工)(95 中山資工)(95 交大電子)(96 中央通訊)(96 輔大資工)(95 中原資工)(95 朝陽資工)(95 高雄電機)(94 中央通訊)(94 北科電通)(94 淡江電機)(94 朝陽資工)(94 中興電機)(93 中原資工)(92 政大資科)(92 台師資工)(92 長庚資工)(91 中央電機)

hazard	意義 (96 朝陽資工) (96 中原資工)	解決方法
structural	同一個 clock cycle 中,硬體(functional unit)不能被完成多個 instruction 的需求,例如一個記憶體，不可在同一 clock 作讀取與寫入的動作	1.stall 2.replicate functional units
Data	一指令之運算元必須參考先前指令的執行結果，但先前指令還在 pipeline 中尚未執行完成	1.stall 2.forwarding (bypassing) 3.instruction reordering
Control	程式下一個要執行的指令必需依據某 condition ,但在 condition 未被算出來時,就決定了程式的下一個指令,則所執行的 instruction 有可能是錯的	1.stall 2.delayed branch 3. branch predict- (a) static – taken/ not taken (b)dynamic – history register

94 暨南資工,93 台大電機

List a example for three hazard

Structural Hazard

```
load $s0, 100($t0)
add   $t0, $t0, $t0
add   $t0, $t0, $t0
add   $t0, $t0, $t0
```

例子:當第 4 個 clock cycle 時,第 1 條指令要從 memory load data, 第 4 條指令要從 memory fetch instruction,就會發生 Structural Hazard

解決:使用 (replicated) memory,有 data memory 和 instruction memory

Data hazard

```
add     $s0, $t0, $t1
sub     $t2, $s0, $t3
```

add 指令的 $s0 未算出結果 , sub 指令就捷取 $s0,此時資料錯誤

解決(1) forwarding or (2)stall

control Hazard example (93 台科電子) (92 長庚資工)

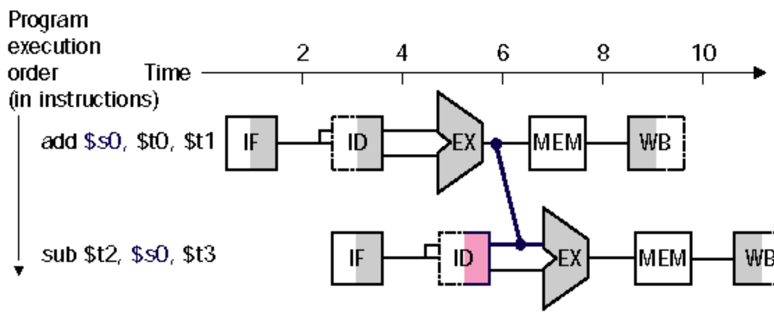
```
            add $s1, $s2, $s3
            beq   $s1, $s4 ,Label
            add $s3, $s3, $s3
LABEL:      add $s3, $s3, $s3
```

例子: beq 時無法得知 s1 的結果,會造成 control Hazard

95 交大電子

Types of Data Hazards and explain
(a)RAW(read after write): 後面指令(j)在前面指令(I)寫入前就讀取 I: add **r1**,r2,r3 J: sub r4,**r1**,r3 (b)WAR(write after read): 後面指令(j)在前面指令(I)讀取前就寫入 不可能發生於 **MIPS 5-stage pipeline**,因 stage 2➔reads, stage 5➔write I: sub r4,**r1**,r3 J: add **r1**,r2,r3 (c)WAW(write after write): 後面指令(j)在前面指令(I)寫入前就寫入 發生於 pipeline 設計寫入不只一個 stage 不可能發生於 **MIPS 5-stage pipeline**,因都是 stage 5➔write I: sub **r1**,r4,r3 J: add **r1**,r2,r3

92 清大資工,算盤習題

Data hazards occur when an instruction depends on the result of a previous instruction still in the pipeline. For example, **add $s0, $t0, $t1** **sub $t2, $s0, $t3** The add instruction does not write its result until the fifth stage, meaning that we should have to add three bubbles to the pipeline. So that the subtraction can compute correctly
 Program execution order (in instructions) Time 2 4 6 8 10 add $s0, $t0, $t1 IF ID EX MEM WB sub $t2, $s0, $t3 IF ID EX MEM WB

Load-use data hazard

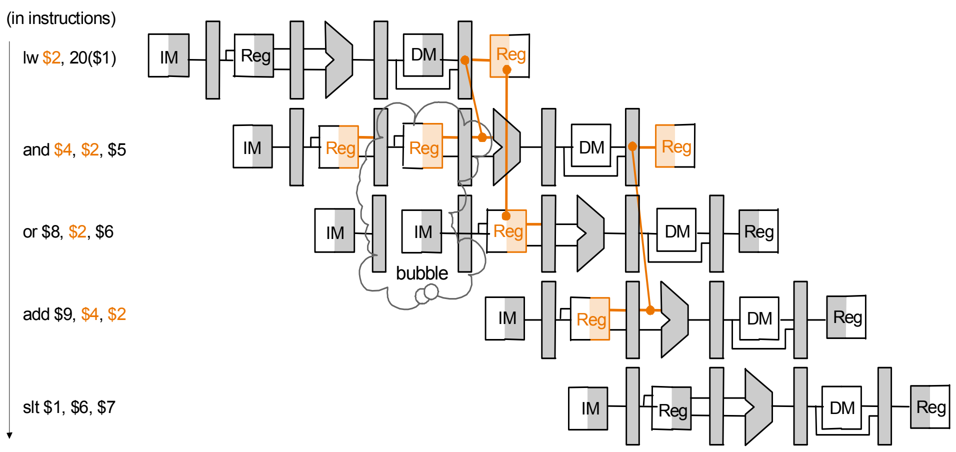
一種特定的 data hazard 形式,當需要 load 指令所 load 出來的資料無法立即獲得

MIPS指令中常發生於一個load指令緊接一個R-type指令

Load-use data hazard

lw	$2,20($1)
and	$4, $2, $5
or	$8,$2,$6
add	$9,$4,$2
slt	$1,$6,$7

solve data hazard by forwarding ,add one stall



95 清大通訊

Reorder the instructions to avoid as many pipelines stalls as possible.

		# reg $2 has the address of v[k]
lw	$15, 0($2)	# reg $15(temp) = v[k]
lw	$16, 4($2)	# reg $16 = v[k+1]
sw	$16, 0($2)	# v[k] = reg $16
sw	$15, 4($2)	# v[k+1] = reg $15 (temp)

lw	$15, 0($2)
lw	$16, 4($2)
sw	**$15, 4($2)**
sw	**$16, 0($2)**

(21) data dependency

長庚資工 95

Identify all data dependences in the following code

```
sub   $1,$1,$3
lw    $2,100($1)
lw    $3,100($2)
add   $4,$2,$3
sw    $4,100($3)
```

sub $1,$1,$3	IF	ID	Exe	Mem	Wb				
Lw $2,100($1)		IF	ID	Exe	Mem	Wb			
Lw $3,100($2)			IF	ID	Exe	Mem	Wb		
add $4,$2,$3				IF	ID	Exe	Mem	Wb	
sw $4,100($3)					IF	ID	Exe	Mem	Wb

中原資工 95

(a)Identify all data dependences in the following code

```
add   $3,$4,$2
sub   $5,$3,$1
lw    $6,200($3)
add   $7,$3,$6
```

(b)which dependences be data hazard will be solve via forwarding ?

(c) which dependences be data hazard will cause stall?

(a) 沒有前半 cycle 寫入,後半 cycle 讀取

add	IF	ID	Exe	Mem	Wb						
sub		IF	ID	Exe	Mem	Wb					
Lw			IF	ID	Exe	Mem	Wb				
add				IF	ID	Exe	Mem	Wb			

(b) I1➔I2 , I1➔I3, I1➔I4 可由 forwarding 解決

(c)I3➔I4 需 stall + forwarding 解決

93 彰師資工

(a) Identify all data dependences in the following code

```
lw      $t0,0($s1)
addu    $t0,$t0,$s2
sw      $t0,0($s1)
addi    $t0,$t0,4
bne     $s1,$zero,Loop
```

(b)which dependences be data hazard will be solve via forwarding ?

(a)

Lw	IF	ID	Exe	Mem	Wb						
add		IF	ID	Exe	Mem	Wb					
addi			IF	ID	Exe	Mem	Wb				
sw				IF	ID	Exe	Mem	Wb			
add					IF	ID	Exe	Mem	Wb		

(b)

(I1,I2) , (I1,I3) solve via forwarding

(I2,I4) solve by **stall** + forwarding

輔大資工 95

(b) Identify all data dependences in the following code

```
lw   $2,0($10)
sub  $4,$2,$3
mult $5,$4,$2
sw   $4,200($5)
add  $3,$1,$2
```

(b)which of upper be data hazard ?

(b)which dependences be data hazard will not be solve via forwarding ?

Lw	IF	ID	Exe	Mem	Wb						
sub		IF	ID	Exe	Mem	Wb					
mult			IF	ID	Exe	Mem	Wb				
sw				IF	ID	Exe	Mem	Wb			
add					IF	ID	Exe	Mem	Wb		

(c) I1➔I2 , I1➔I3, I2➔I3 , I2➔I4 , I3➔I4

(d) I1➔I2 ,load-use 需 stall + forwarding 才可 solve

交大資工 96-10

What is the average CPI for each of the following 4 schemes taking to execute the code sequence below? (Note: For the pipeline scheme, there are five stages: IF, ID, EX, MEM, and WB. We assume the reads and writes of register file can occur in the same clock cycle, and the stall circuits are available.)

```
add $t3, $s1, $s2
sub $t1, $s1, $s2
lw   $t2, 100($t3)
sub $s1, $t1, $t2
```

(a) single cycle scheme;

(b) multi-cycle scheme without pipelining;

(c) pipelined scheme without data forwarding hardware;

(d) pipelined scheme with data forwarding hardware (one from EX/MEM to ALU input, and the other from MEM/WB to ALU input) available.

(a) CPI = 1

(b) CPI = (4 + 4 + 4 + 4)/4 = 4

(c) The clocks for executing this code = (5 – 1) + 4 + 3 = 11, CPI = 11/4 = 2.75

```
add $t3, $s1, $s2
sub $t1, $s1, $s2
stall (解決 I1,I3 的 $t3 data hazard )
lw   $t2, 100($t3)
stall
stall (解決 I4,I5 的 $t2 load-use data hazard )
sub $s1, $t1, $t2
```

(d) The clocks for executing this code = (5 – 1) + 4 + 1 = 9, CPI = 9/4 = 2.25

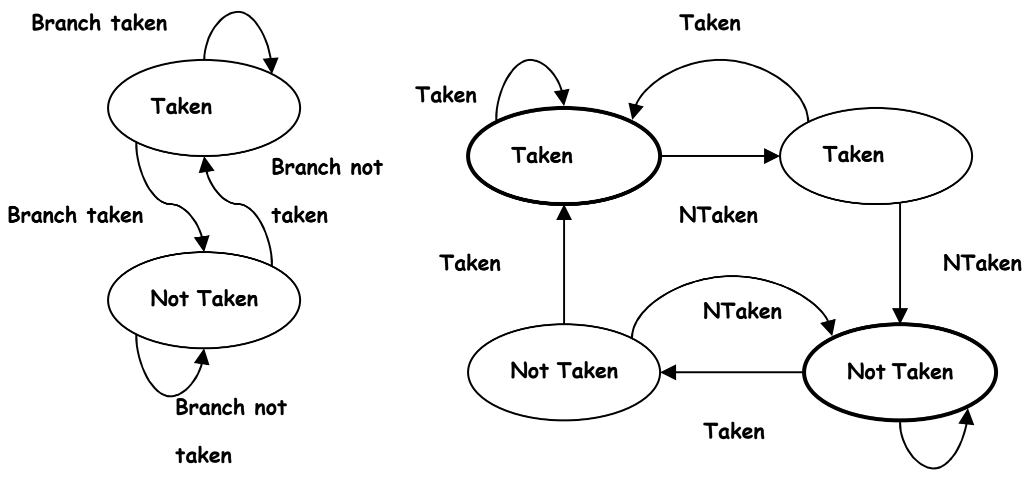
```
    add $t3, $s1, $s2
sub $t1, $s1, $s2
forwarding (解決 I1,I3 的 $t3 data hazard )
lw   $t2, 100($t3)
forwarding + stall(解決 I4,I5 的 $t2 load-use data hazard )
sub $s1, $t1, $t2
```

(22) branch prediction

97 中正資工,95 清大資工,94 彰師電子,**93 台大電機**:2 bit,91 中央資工,90 台大資工

(1)what are static and dynamic branch prediction?

(2)compare the advantage and disadvantage for static and dynamic branch prediction



	Static prediction	Dynamic prediction
explain	預測分支總是成立(taken)或總是不成立(not taken)	1 bit or 2 bit,預測在 run time 時會由之前的 run time information 做改變
advantage	實作容易,Hardware cost 低	猜對機率高,效能高
disadvantage	猜錯機率高,效能低	實作不易,Hardware cost 高

97 台科大電子 Check yourself

Consider three branch prediction schemes: branch not taken, predict taken, and dynamic prediction.Assume that they all have zero penalty when they predict correctly and 2 cycles when they are wrong. Assume that the average predict accuracy of the dynamic predictor is 90%. Which predictor is the best choice for the following branches?

1. A branch that is taken with 5% frequency
2. A branch that is taken with 95% frequency
3. A branch that is taken with 70% frequency

1. Prediction not taken. 2. Prediction taken. 3. Dynamic prediction

96 中山資工

Reordering the code (pipeline scheduling or instruction scheduling) Example A = B + E C = B + F ---	
lw $t1, 0($t0) lw $t2, 4($t0) add $t3, $t1, $t2 sw $t3, 12($t0) lw $t4, 8($t0) add $t5, $t1, $t4 sw $t5, 16($t0)	lw $t1, 0($t0) lw $t2, 4($t0) **lw $t4, 8($t0)** add $t3, $t1, $t2 sw $t3, 12($t0) add $t5, $t1, $t4 sw $t5, 16($t0)

branch delay slot (compiler 在做) 93 交大資工

branch delay slot 中的 instruction 由哪一個 instruction 填入

(a)branch instruction 的前一個指令

(b)條件跳躍目標指令 ➔把條件跳躍目標指令複製**(duplicated)**進 branch delay slot

(c) 條件跳躍不成立的下一個指令

當(a)行不通時 才考慮(b)(c),(a)的 performance 最高

Scheduling Strategy	Requirements	Improve Performance When?
From before	Branch 與重排的那個 instruction 沒有相依性	**Always**
From target (loop 多使用此法)	**重排的 instruction 為條件成立的下一個 instruction,** 當 **branch is not taken(**不成立**)時**,必需重排指令(rescheduled instructions),複製指令	**When branch is taken(**成立**)**因複製指令,程式碼變大
From fall through	**重排的 instruction 為條件不成立的下一個 instruction,** 當 **branch is taken(**成立**)時**,必需重排指令(rescheduled instructions)	**When branch is not taken (**不成立**)**

台大資工 94-05

Assume an instruction set that contains 5 types of instructions: load, store, R-format, branch and jump. Execution of these instructions can be broken into 5 steps: instruction fetch, register read, ALU operations, data access, and register write. Table 1 lists the latency of each step assuming perfect caches.

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write
Load	2ns	1ns	1ns	2ns	1ns
Store	2ns	1ns	1ns	2ns	
R-format	2ns	1ns	1ns		1ns
Branch	2ns	1ns	1ns		
Jump	2ns				

(a) What is the CPU cycle time assuming a multicycle CPU implementation (i.e., each step in Table 1 takes one cycle)?

(b)assuming the instruction mix shown below,what is the average CPI of the multicycle provessor without pipeline?Assume that the I-cache and D-cache miss rate are 3% and 10%,and the cache miss panality is 12 CPU cycle

Frequency	Instruction fetch
Load	40%
Store	30%
R-format	15%
Branch	10%
Jump	5%

(c)To reduce the cache miss rate ,the architecture team is considering increasing the data cache size.They find that by doubling the cache size,they can eliminate half of data cache misses.However,the data access stage now take 4ns.Do you suggest them to double the data cache size? Explain your answer

(a) 取最長執行步驟的時間作爲 CPU cycle time 因此 multicycle implementation 的 CPU cycle time 爲 2ns

(b) Average CPI=(5*0.4+4*0.3+4*0.15+3*0.1+1+0.05=4.15)+3%*12+(40%+30%)*10%*12 =5.35

(c) Average CPI=(5*0.4+4*0.3+4*0.15+3*0.1+1+0.05=4.15)+3%*12+(40%+30%)***5%***12 =4.93

4.93*4=19.72 比原本執行時間長,因此不建議

3th only (96 中山資工) (97 中山資工:4.12)

200 ps for memory access, 100 ps for ALU operation, 50 ps for register file read or write

25% loads, 10% stores, 52% ALU, 11 % branches, 2% jumps

Clock cycle➔ loads=5, stores=4, ALU=4, branches=3, jumps=3

CPI➔loads=1.5, stores=1, ALU=1, branches=1.25, jumps=2

Compare performance for single-cycle, multicycle, pipeline

For single-cycle machine:

CPI = 1

Clock cycle time = 200 + 50 + 100 + 200 + 50 = 600 ps

Average instruction time = 1 × 600 = 600 ps

For multicycle machine:

CPI = 0.25 × 5 + 0.1 × 4 + 0.52 × 4 + 0.11 × 3 + 0.02 × 3 = 4.12

Clock cycle time = 200 ps

Average instruction time = 4.12 × 200 = 824 ps

For pipeline machine:

Effective CPI = 1.5 × 0.25 + 1 × 0.1 + 1 × 0.52 + 1.25 × 0.11 + 2 × 0.2 = 1.17

Clock cycle time = 200 ps

Average instruction time = 1.17 × 200 = 234 ps

The relative performance of the three machines is **Pipeline > single-cycle > multi-cycle** in terms of average instruction time.

Check yourself (96 台科電子) (94 大同資工)

State whether the following techniques or components are associated primarily with a software- or hardware-based approach to exploiting ILP. In some cases, the answer may be both.

1. Branch prediction 2. Multiple issue 3. VLIW 4. Superscalar
5. Dynamic scheduling 6. Out-of-order execution 7. Speculation 8. EPIC
9. Reorder buffer 10. Register renaming 11. Predication

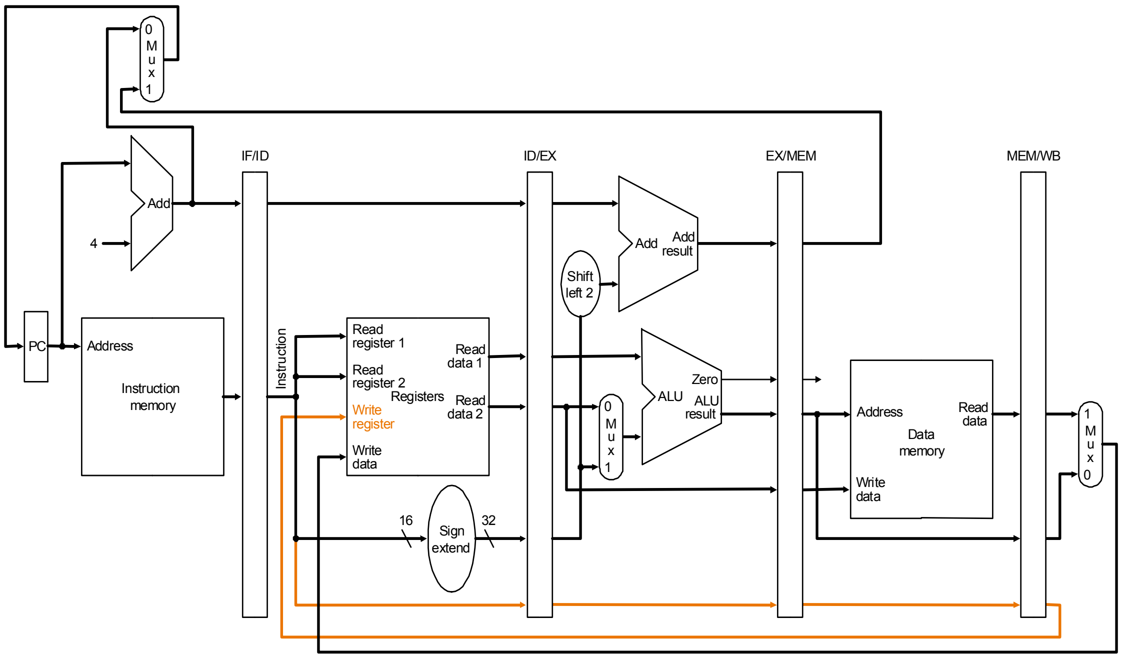
(1) B (2)B (1)(2)皆有static and dynamic (3)S

(4)H (5)H (6)H (7) B 使用cpu or processor 猜測

(8) B (9) H (10) B (11)S

中山資工 96-05-02

For each pipeline register in Fig. 1, label each portion of the pipeline register with the name of the value that is loaded into the register. Determine the length of each field in bits. For example, the IF/ID pipeline register contains two fields, one of which is an instruction field that is 32 bits wide.



Pipeline register	IF/ID	ID/EX	EX/MEM	MEM/WB
	PC + 4 (32 bits)	PC + 4(32 bits)	Branch target address(32 bits)	Memory data(32 bits)
	Instruction (32 bits)	Register data 1(32 bits)	Zero indicator (1 bit)	ALU result (32 bits)
		Register data 2(32 bits)	ALU result (32 bits)	Destination register No. (5 bits)
		Sign-extension unit output(32 bits)	Register data 2 (32 bits)	
		Register No. rt(5 bits)	Destination register(5 bits)	
		Register No. rd(5 bits)		
Total (bits)	**64**	**138**	**102**	**69**

中央電機 93

A pipeline has four stage ,need time be 80ns,50ns,80ns,and 80ns (a)How much time is the pipeline execute 10 instruction? (b)How much time is the pipeline execute 100 instruction?
$(a)80*(4+10-1)=1040$ $(b)80*(4+100-1)=8240$

中央通訊 93

A non-pipeline has five stage ,need time be 50ns,50ns,60ns,60ns,and 50ns (a)what is the instruction latency for this machine? (b)How much time if it execute 100 instruction? (c)How much time if using pipeline execute 100 instruction (stage latency=5ns)?
$(a)50+50+60+60+50=270ns$ $(b)270*100=27000ns$ $(c)(60+5)*(5+100-1)=6760$

中央資工 96-01

There is an unpipelined processor that has a 1 ns clock cycle and that uses 4 cycles, for ALU and branch operations and 5 cycles for memory operations. Assume that the relative frequencies of these operations are 40%, 20%, and 40%, respectively. Suppose that due to clock skew and setup, pipelining the processor adds 0.2 ns of overhead to the clock. Ignoring any latency impact, how much speedup in the instruction execution rate will we gain from a pipeline implementation?
Average instruction execution time = 1 ns × ((40% + 20%) × 4 + 40% × 5)= 4.4ns Speedup from pipeline = Average instruction time unpiplined/Average instruction time pipelined = 4.4ns/1.2ns = 3.7

94 中興電機

Sub $3,$4,$5 Lw $4,10($6) Add $7,$3,$4 Sw $7,20($6) Add $7,$2,$1 How many cycle in (a)single cycle machine (b)multi-cycle machine (c)pipeline		
(a)5*5=25	(b)4+5+4+4+4=21	(c) 5+5-1=9

(23) cache address mapping

94 北科電通,94 輔大電子,93 清大電機 算盤習題

Consider a cache with 64 blocks and a block size of 16 bytes. What block number does byte address 1200 map to?

$\left\lfloor \frac{1200}{16} \right\rfloor$=75 , 75 % 64=11

96成大資工,96高雄電機,95高雄電機 算盤習題

Assume there are three small caches, each consisting of four one-word blocks. One cache is fully associative, a second is two-way set associative, and the third is direct mapped. Find the number of misses for each cache organization given the following sequence of block addresses: 0, 8,0,6,8.

The direct-mapped cache: 5 miss

Block address	**Cache block**	**Hit or miss**	**Contents of cache blocks after reference**			
			0	1	2	3
0	0	Compulsory miss	**Mem[0]**			
8	0	Conflict miss	**Mem[8]**			
0	0	Conflict miss	**Mem[0]**			
6	2	Compulsory miss	Mem[0]		**Mem[6]**	
8	0	Conflict miss	**Mem[8]**		Mem[6]	

Two way set associative: 4 miss

Block address	**Cache block**	**Hit or miss**	**Contents of cache blocks after reference**				
			Set 0	Set 0	Set 1	Set 1	
0	0(last bit)	Compulsory miss	**Mem[0]**				
8	0	Compulsory miss	Mem[0]	**Mem[8]**			
0	0	hit	Mem[0]	Mem[8]			
6	**1**	Compulsory miss	Mem[0]	**Mem[6]**			
8	0	Conflict miss	**Mem[8] (LRU)**	Mem[6]			

fully associative: 3 miss

Block address	**Cache block**	**Hit or miss**	**Contents of cache blocks after reference**			
			block 0	block 1	block 2	block 3
0	0	Compulsory miss	**Mem[0]**			
8	0	Compulsory miss	Mem[0]	**Mem[8]**		
0	0	hit	Mem[0]	Mem[8]		
6	2	Compulsory miss	Mem[0]	Mem[8]	**Mem[6]**	
8	0	hit	Mem[0]	Mem[8]	Mem[6]	

94 交大電控

4 one-word blocks ,address in order 0,8,0,10,2,0,8,0,10,2

(a)direct map (b)two way (LRU) (c)fully

Address	**Block #**	**Hit/miss**	**0**	**1**	**2**	**3**
0000	00	complusory	**Mem[0]**			
1000	00	comflict	**Mem[8]**			
0000	00	comflict	**Mem[0]**			
1010	10	complusory	Mem[0]		**Mem[10]**	
0010	10	comflict	Mem[0]		**Mem[2]**	
0000	00	hit	Mem[0]		Mem[2]	
1000	00	comflict	Mem[0]		**Mem[10]**	
0000	00	hit	Mem[0]		Mem[10]	
1010	10	hit	Mem[0]		Mem[10]	
0010	10	comflict	Mem[0]		**Mem[2]**	
Address	**set #**	**Hit/miss**	**0**		**1**	
0000	0	complusory	**Mem[0]**			
1000	0	complusory	Mem[0]	**Mem[8]**		
0000	0	hit	Mem[0]	Mem[8]		
1010	0	conflict	Mem[0]	**Mem[10]**		
0010	0	conflict	**Mem[2]**	Mem[10]		
0000	0	conflict	Mem[2]	**Mem[0]**		
1000	0	conflict	**Mem[8]**	Mem[0]		
0000	0	hit	Mem[8]	Mem[0]		
1010	0	conflict	**Mem[10]**	Mem[0]		
0010	0	conflict	Mem[10]	**Mem[2]**		
Address	**Block #**	**Hit/miss**	**0**	**1**	**2**	**3**
0000	00	complusory	**Mem[0]**			
1000	00	complusory	Mem[0]	**Mem[8]**		
0000	00	hit	Mem[0]	Mem[8]		
1010	10	complusory	Mem[0]	Mem[8]	Mem[10]	
0010	10	complusory	Mem[0]	Mem[8]	Mem[10]	Mem[2]
0000	00	hit	Mem[0]	Mem[8]	Mem[10]	Mem[2]
1000	00	hit	Mem[0]	Mem[8]	Mem[10]	Mem[2]
0000	00	hit	Mem[0]	Mem[8]	Mem[10]	Mem[2]
1010	10	hit	Mem[0]	Mem[8]	Mem[10]	Mem[2]
0010	10	hit	Mem[0]	Mem[8]	Mem[10]	Mem[2]

清大資工 94-09

Consider the following sequence of address reference given as word address:
22, 10, 26, 30, 18, 10, 14, 30, 11, 15, 19
For a 2-way set associative cache with a block size of 8 bytes ,a word size of 4 bytes, a data capacity of 64bytes and the LRU replacement, label each reference in the sequence as a hit or a miss. Assume that the cache is initially empty.

a word size of 4 bytes➔存 1 個 number need 4 bytes

a block size of 8 bytes➔1 個 block 可存 2 個 word (所以最後 1bit 不能當 set number)

如果此題為 1 個 block=1 個 word 則直接取 後面為 set num

但此題 1 個 block 可存 2 個 word ➔所以先 mode 2

total block➔64/8=8 ,2-way ➔one way has 4 sets

Address (decimal)	Address (binary)	tag	Set index	Hit/Miss
22	010110	010	11	Compulsory Miss
10	001010	001	01	Compulsory Miss
26	011010	011	01	Compulsory Miss
30	011110	011	11	Compulsory Miss
23	010111	010	11	**Hit**
18	010010	010	01	Conflict Miss
10	001010	001	01	Conflict Miss
14	001110	001	11	Conflict Miss
30	011110	011	11	Conflict Miss
11	001011	001	01	**Hit**
15	001111	001	11	**Hit**
19	010011	010	01	**Hit**

	set 0	set 1	set 2	set 3
Block 0		Mem[10] [11] Mem[18] [19]		Mem[22] [23] Mem[30] [31]
Block 1		Mem[25] [26] Mem[10] [11]		Mem[30] [31] Mem[14] [15]

清大通訊 94-01

1,4,8,5,20,17,19,56,9,11,4,43,5,6,9,17.show the this and final cache contents for a two-way set associative cache with one-word blocks and a total size of 16 words. Assume LRU replacement.

16 words,2 way ➔ one way has 8 sets

Address (decimal)	Address (binary)	tag	Set (index)	Hit/Miss
1	000**001**	0	1	Compulsory Miss
4	000**100**	0	4	Compulsory Miss
8	001**000**	1	0	Compulsory Miss
5	000**101**	0	5	Compulsory Miss
20	010**100**	2	4	Compulsory Miss
17	010**001**	2	1	Compulsory Miss
19	010**011**	2	3	Compulsory Miss
56	111**000**	7	0	Compulsory Miss
9	001**001**	1	1	Conflict Miss
11	001**011**	1	3	Compulsory Miss
4	000**100**	0	4	**Hit**
43	101**011**	5	3	Miss
5	000**101**	0	5	**Hit**
6	000**110**	0	6	Compulsory Miss
9	001**001**	1	1	**Hit**
17	010**001**	2	1	**Hit**

	set 0	set 1	Set 2	set 3	set 4	set 5	set 6	set 7
Cell 0	Mem[8]	Mem[1] Mem[9]		Mem[19] Mem[43]	Mem[4]	Mem[5]	Mem[6]	
Cell 1	Mem[56]	Mem[17]		Mem[11]		Mem[20]		

交大資工 96-08

Assume the three caches below, each consisting of 16 words.
address references addresses: 2, 3, 4, 16, 18, 16, 4, 2. LRU used
(a) a direct-mapped cache with 16 one-word blocks;
(b) a direct-mapped cache with 4 four-word blocks;
(c) a four-way set associative cache with block size of one-word.

(a) **18是第一次出現,雖然cache中有 data (2), 但還是算 compulsory miss**

Address	tag	index	Hit/miss	3C`s miss
2 (00010)	0	0010	Miss	compulsory
3 (00011)	0	0011	Miss	compulsory
4 (00100)	0	0100	Miss	compulsory
16 (10000)	1	0000	Miss	compulsory
18 (10010)	1	0010	Miss	**compulsory**
16 (10000)	1	0000	hit	
4 (00100)	0	0100	hit	
2 (00010)	0	0010	miss	conflict

(b)**因一個block 含4個wod ,所以先mode 4 完了,LSB 的2個bit才爲set num**

Address	tag	index	Hit/miss	3C`s miss
2 (0**00**10)	0	00	Miss	compulsory
3 (0**00**11)	0	00	hit	
4 (0**01**00)	0	01	Miss	compulsory
16 (1**00**00)	1	00	Miss	**compulsory**
18 (1**00**10)	1	00	hit	
16 (1**00**00)	1	00	hit	
4 (0**01**00)	0	01	hit	
2 (0**00**10)	0	00	miss	conflict

(c)

Address	tag	index	Hit/miss	3C`s miss
2 (00010)	000	10	Miss	compulsory
3 (00011)	000	11	hit	
4 (00100)	001	00	Miss	compulsory
16 (10000)	100	00	Miss	compulsory
18 (10010)	100	10	Miss	compulsory
16 (10000)	100	00	hit	
4 (00100)	001	00	hit	
2 (00010)	000	10	hit	

96 雲科資工　93 中央電機　93 彰師資工

Here is a series of address references given as word addresses:1,4,8,5,20,17,19,56,9,11,4,43,5,6,9,17

(1) assuming a direct-mapped cache with 16 one-word blocks that initially empty,label each reference in the list as a hit or miss,show the final contents of the cache

(2) show the hit or misses and final cache content for a direct-mapped cache with four-word blocks and a total size of 16 words

(3) show the hits or misses and final cache content for a two way associative cache with one-word blocks and a total size of 16 words,assume LRU

(4) what the hit ratios of above question

(1)

1	4	8	5	20	17	19	56	9	11	4	43	5	6	9	17
m	m	m	m	m	m	m	m	m	m	m	m	h	m	h	h

cache

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1 17		19	4 20 4	5	6		8 56	9		11 43				

(2)先 mod 16 ,再 mod　4

1	4	8	5	20	17	19	56	9	11	4	43	5	6	9	17
m	m	m	h	m	m	h	m	m	h	m	m	h	h	m	h

0	1	2	3
0,1,2,3 16,17,18,19	4,5,6,7 20,21,22,23 4,5,6,7	8,9,10,11 56,57,58,59 8,9,10,11 40,41,42,43 8,9,10,11	

(3)先 mod 16　0~7➔ way 1　, 8~15➔ way 2

1	4	8	5	20	17	19	56	9	11	4	43	5	6	9	17
m	m	m	m	m	m	m	m	m	m	h	m	h	m	h	h

Way/set	0	1	2	3	4	5	6	7
0	1	4	5	20	17	19	56	6
1	8	9	11	43				

(24) cache size calculation

95雲科電機　93 清大電機　算盤習題

Assuming a cache of 4K blocks, a four-word block size, and a 32-bit address, find the total number of sets and the total number of tag bits for caches that are direct mapped, two-way and four-way set associative, and fully associative
4-word (16 byte) per block, total (32 – 4) = 28 bits for tag and index **1. Direct mapped:** number of sets = 4K, Log2(4K) = 12, number of tag bits, (28 – 12) × 4K = 64Kbits **2. 2-way associative:** number of sets = 4K/2 = 2K, number of tags = (28 – 11) × 2 × 2K = 34 × 2K = 68Kbits **3. 4-way associative:** number of sets = 4K/4 = 1K sets, number of tags = (28 – 10) × 4 × 1K = 72 × 1K = 72Kbits **4. Fully associative:** Only one set, number of tags = 28 × 4K = 112Kbits

94成大資工,95 清大電機,94 政大資科,94 大同資工,93彰師資工 94大同資工,91中正資工

Assume the following : (1)k is the number of bits for a computer`s address size (using Byte addressing) (2)S is cache size in byte (3)B is block size in bytes ,B=2^b (4)A stands for A-way associative cache Figure out the following quantities in terms of S,B,A,and k: (a) the number of sets in the cache (b) the number of index bits in the address (c) the number of bits needed to implement the cache
Address size: k bits　　　　Cache size: S bytes/cache Block size: B = 2^b bytes/block　　　Associativity: A blocks/set Number of sets in the cache = $\frac{S}{AB}$ Number of bits for index = $\log_2\left(\frac{S}{AB}\right)=\log_2\left(\frac{S}{A}\right)-b$ Number of bits for the tag = $k-\left(\log_2\left(\frac{S}{A}\right)-b\right)-b=k-\log_2\left(\frac{S}{A}\right)$ Number of bits needed to implement the cache =sets/cache × associativity × (data + tag + valid) = $\frac{S}{AB}*A*\left(8*b+k-\log_2\left(\frac{S}{A}\right)+1\right)=\frac{S}{B}*\left(8*b+k-\log_2\left(\frac{S}{A}\right)+1\right)bits$

95 中原資工 ，95 元智資工

Suppose that in 1000 memory reference there are 60 misses in the first-level cache, 30 misses in the second-level cache, 5 misses in the third-level cache.Assume the miss penality from the L3 cache is 10 clocks, the hit time of the L2 cache is 5 clocks, the hit time of the L1 cache is 1 clocks,and there are 1.5 memory reference per instruction (a) what`s the global miss rate for each level of caches? (b) what`s the local miss rate for each level of caches? (c) What is the average memory access time? (d) What is the average stall cycle per instruction ?
$(a) L1: \frac{60}{1000} = 6\%, L2: \frac{30}{1000} = 3\%, L3: \frac{5}{1000} = 0.5\%$ $(b) L1: \frac{60}{1000} = 6\%, L2: \frac{30}{60} = 50\%, L3: \frac{5}{30} = 16.7\%$ $(c) 1 + 6\% * 5 + 3\% * 10 + 0.5\% * 100 = 2.1$ clock cycle $(d) 2.1 - 1 = 1.1$ stall cycle

暨南資工 92

Suppose that in 1000 memory reference there are 50 misses in the first-level cache, 20 misses in the second-level cache, 5 misses in the third-level cache (a)what are the carious miss rate ?.Assume the miss penality from the L3 cache is 100 clocks, the hit time of the L3 cache is 10 clocks,the hit time of the L2 cache is 4 clocks, the hit time of the L1 cache is 1 clocks,and there are 1.2 memory reference per instruction What is the average memory access time?
法一　Gobal ➔L1: 5% , L2: 2% , L3: 0.5% 1+0.05*4+0.02*10+0.005*100=1.9 cycle 法二　local ➔L1: 5% , L2: 40% , L3: 25% 1+0.05*(4+0.4*(10+0.25*100))= 1.9 cycle

94 交大資工　3 版習題

(1)How faster a processor would run with a perfect cache?

— Instruction cache miss rate for a program: 2%

— Data cache miss rate: 4%

— Processor CPI: 2 without any memory stall

— Miss penalty: 40 (100) cycles for all misses

Load & Store = 36%

(2)

If the processor CPI is reduced from 2 to 1, what will happen in the previous example?

(3)

If the processor clock rate doubled, what will happen in the previous example? Assuming that the absolute time is not changed to handle a cache miss

---

(1)

Instruction miss cycle = I × 2% ×100 = 2I

Data miss cycle = I × 36% × 4% × 100 = 1.44 I

Total memory stall cycle = 2I + 1.44 I = 3.44 I

CPI with memory stall = 2 +3.44 = 3.365.44

Ration of the execution time = 5.44/2 = 2.72

perfect cache performance 校能好了 2.72倍

(2)

System with cache miss, CPI = 1 +3.44=4.44

The system with the perfect cache will be 4.44/1 = 4.44 faster

The amount of execution time spent on memory stall will rise from

3.44/5.44= 63 % to 3.44/4.44 = 77%

Memory stall cycle 的比例

(3)

Measured in faster clock cycle, the new miss penalty will be 200 cycles

Total miss cycles per instruction = 2% × 200 + 36% × (4% × 200) = 6.88

Faster system with cache miss, CPI = 2 + 6.88= 8.88

Slower system with cache miss, CPI = 5.44

The slower clock system will be

Execution time of slow clock/Execution time of faster clock

= I × CPI_slow × Cycle time / (I × CPI_fast × $\frac{1}{2}$ × Cycle time)

= 5.44/ (8.88 × $\frac{1}{2}$) = 1.23 faster

97 交大資工 3 版習題 95 中山資工類似

Consider three machines with different cache configurations：

Cache 1：Direct-mapped with **one-word blocks**.

Cache 2：Direct-mapped with **four-word blocks.**

Cache 3：Two-way set associative with **four-word blocks.**

The following miss measuremented have been made：

Cache 1：Instruction miss is 4%, data miss rate is 6%.

Cache 2：Instruction miss is 2%, data miss rate is 4%.

Cache 3：Instruction miss is 2%, data miss rate is 3%.

For these machines, one-half of the instructions contain a data reference. Assume that the cache miss penalty is 6+Block size in words. The CPI for this workload was measured on a machine with cache 1 and found to be 2.0.

(1) Determine which machine spends the most cycles on cache misses

(2) Determine the CPI for process with cache 2 and cache 3 respectively

(3)assume the cycle time for the first and second processor are 420ps,and 310 ps for third processor, determine which process is faster and which is slowest

(1)C1

Cache	Miss penalty	I cache miss	D cache miss	Total Miss
C1	6 + 1 = 7	4% × 7 = 0.28	6% × 7 = 0.42	0.28 + 0.42/2 = 0.49
C2	6 + 4 = 10	2% × 10 = 0.2	4% × 10 = 0.4	0.2 + 0.4/2 = 0.4
C3	6 + 4 = 10	2% × 10 = 0.2	3% × 10 = 0.3	0.2 + 0.3/2 = 0.35

C1 miss cycle 最長,

(2)

CPI_base = CPI – CPImisses= 2 – 0.49 = 1.51

CPI with cache 2=1.51+0.4=1.91

CPI with cache 3=1.51+0.3=1.86

(3)

Execution Time=CPI*Clock cycle time*Instruction count(IC)

Execution Time for C1 = 2×420 ps×IC = 8.4×10^{-10}×IC

Execution Time for C2 =(1.51+0.4)×420 ps×IC = 8.02×10^{-10}×IC

Execution Time for C3 =(1.51+0.35)×310 ps×IC = 5.77×10^{-10}×IC

C3 最快.C1 最慢

92清大電機,94清大通訊　數字

Cache 1：I　miss is 4%, d miss is 8%.　10ns

Cache 2：I　miss is 2%, d miss is 5%.　10ns

Cache 3：I　miss is 2%, d miss is 4%.　12ns

Ans:C1　,20×10^{-9}×IC, 18.9×10^{-9}×IC, 22.1×10^{-9}×IC

台大資工 95-03

Suppose you had a computer hat, on average, exhibited the following properties on the programs that you run:

Instruction miss rate: 2% Data miss rate: 4%

Percentage of memory instructions: 30% Miss penalty: 100 cycles

*Option #1: Get a new processor that is twice as fast as your current computer. The new processor's cache is twice as fast too, so it can keep up with the processor.

* Option #2: Get a new memory that is twice as fast. Which is a better choice? And what is the speedup of the chosen design compared to the old machine?

Option 2 is $4.2/2.6 = 1.62$ times faster than the old machine.

Suppose that the base CPI = 1

$CPIold = 1 + 0.02 \times 100 + 0.04 \times 0.3 \times 100 = 4.2$

$CPIopt1 = 0.5 + 0.02 \times 100 + 0.04 \times 0.3 \times 100 = 3.7$

$CPIopt2 = 1 + 0.02 \times 50 + 0.04 \times 0.3 \times 50 = 2.6$

(25) Effective Memory Access time

96 中央資工

A computer system has L1 and L2 caches. The local hit rates for L1 and L2 are 95% and 80%, respectively. The miss penalties are 8 and 60 cycles, respectively. (a) Assume a CPI (Cycles per Instruction) of 1.2 without any cache miss and an average of 1.1 memory accesses per instruction, what is effective CPI after cache misses are factored in? (b) Taking the two levels of caches as a single cache memory, what are its miss rate and miss penalty?

(a) Effective CPI = $1.2 + 1.1 \times [(1 - 0.95)\times 8+(1-0.95)\times(1-0.8)\times 60] = 2.3$

(b) Hit rate = $95\% + 5\% \times 80\% = 99\% \rightarrow$ Miss rate = $1 - 99\% = 1\%$

Total miss penalty 60 cycles

95 中央電機

A computer system has L1 and L2 caches. The local hit rates for L1 and L2 are 90% and 80%, respectively. The miss penalties are 10 and 50 cycles, respectively. Assuming a CPI of 1.2 without any cache misses and an average of 1.1 memory accesses per instruction:

(1) what is the effective CPI after cache misses are factored in?

(2) Taking the two levels of caches as a single cache memory, what are its miss rate and miss penalty?

(1) The effective CPI = $1.2+1.1\times(0.1\times 10 + 0.1\times 0.2\times 50) = 3.4$

(2) Hit rate = $0.9 + 0.1 \times 0.8 = 0.98 \rightarrow$ Miss rate = $1 - 0.98 = 0.02 = 2\%$

Miss penalty = 50 cycles

94 交大資工　3 版習題

Suppose we have a processor with a base CPI 1.0,assuming all reference hit in the primary cache,and a clock rate of 5GHz. Assume a main memory access time of 100 ns,including all the miss handling.Suppose the miss rate per instruction at the primary cache is 2%.How much faster will the processor be if we add a secondary cache that has a 5ns access time for either a hit or miss and is large enough to reduce the miss rate to main memory to 0.5%？

Miss rate 沒說明就用 global miss rate
main memory miss penality 100/0.2=500 clock cycle , one level CPI=1.0+500*0.2%=11.0
The miss penality for an access to the second-level cache is 5/0.2=25 clock cycle
For the two level cache,total CPI=1.0+2%*25+0.5%*500=4.0
The processor with the secondary cache is faster by 11.0/4.0=2.8

93 清大電機

Suppose we have a processor with a base CPI (clock-cycle per instruction) of 1.0, assuming all reference hit in the primary cache, and a clock rate of 500 MHz. Assume a main memory access time of 100 ns, including all the miss handling. Suppose the miss rate per instruction at the primary cache is 5%.
(1) What is the miss penalty to main memory in clock cycles and the effective CPI for this one-level caching processor?
(2) What will the effective CPI and how much faster will the machine be if we add a secondary cache that has a 10-ns access time for either a hit or a miss and the secondary cache is large enough to reduce the miss rate to main memory to 2%?

(1) CPU clock cycle time=1/500MHz=2 ns,
main memory Miss penalty=100/2=50 clock cycles　, CPI = 1 + 50 × 0.05 = 3.5
(2) Miss penalty for second level cache = 10 / 2 = 5 clock cycles
CPI = 1 + 0.05 × 5 + 0.02 × 50 = 2.25　　Speedup = 3. 5 / 2.25 = 1.56

96 中山資工

Suppose we have a processor with a base CPI of 1.0, assuming all references hit in the primary cache, and a clock rate of 5 GHz. Assume a main memory access time of 100 ns, including all the miss handling. Suppose the miss rate per instruction at the primary cache is 2%. How much faster will the processor be if we add a secondary cache that has a 5 ns access time for either a hit or a miss and is large enough to reduce the miss rate to main memory to 0.6%?

main memory miss penalty is 500 clock cycles , one level CPI = 1.0 + 500 × 2% = 11.0
second cache miss penalty= 25 clock cycles , two-level CPI = 1.0+2%×25+0.6%×500=4.5
Thus, the processor with the secondary cache is faster by 11.0/4.5 = 2.44

(26) set associative

Increase the associative in cache will affect
Decrease miss rate **increase hit time** **increase tag bit** **Increase comparators** **increase complexity of LRU implement**

(27) 3Cs Miss

(94台大資工solution)，(92台大資工solution) (95中山資工) (95東華資工定義) (92暨南資工)

Miss	**定義**	**solution**
Compulsory	又稱 **cold start miss,**資料還沒出出現於**cache**中的第一次存取	**↑cache size (associativity)**
Capacity	當**cache**無容納一次**request**的所有**block**	**↑block size**
Conflict	當**set-associative or direct-mapped cache scheme**中, 不同的**data map** 到同一個**cache location**	**↑associativity**

Memory Hierarchy Design Challenges

(96中央資工) (96高應電機) (96長庚資工)(95東華資工)

Design change	**Effects on miss rate**	**Possible effects**
Cache size ↑	**capacity miss ↓**	**access time ↑**
associativity ↑	**conflict miss ↓**	**access time ↑**
block size ↑	**miss rate↓**	**miss penalty ↑**

97 台大資工

Cache:virtually-addressed but physically-tagged,8KB,direct-mapped 32B block size TLB:fully associative,32entries 32-bit physical address Virtual memory:8KB page,64-bit virtual address (a)How many bits are needed in each tag,index,and block offset fields of the cache? (b)how many bits are needed in each tag field of the TLB? (c)why do we want to use a virtually-addressed but physically-tagged cache?
(a) cache 總寬度爲 physical memory 寬度 block offset→5bits , 8KB/32B=2^8,index→8bits ,tag=32-8-5=19 bits (b) $2^{64}/8K=2^{51}$, tag field of the TLB=51 bits (v)Virtually indexed, physically tagged 以 Virtual address 做索引,使用 physical address 當標籤 虛擬索引獲得效能的好處,及從physical addressed cache獲得架構設計簡化的好處 不會有別名的問題(virtual indexed , virtual tagged有 Aliasing的問題)

(28) TLB

(96 中央資工) (93 台大電機)(95 大同資工)(95 朝陽資工)

(93 政大資科)(93 台師資工) (93 大同資工) (92 台科電機)(91 成大電機)

(a)what is TLB?

(b)does TLB miss imply page fault?

(a) Page table 的一個 cache

用來記錄最近記憶體 address mapping 的 cache,爲了降低存取 page table 次數

提高 virtual address 與 physical address 的轉換效率

TLB欄位 ➔Tag ,physical page number ,reference bit ,dirty bit, valid bit

(b) 如果 TLB miss,先判斷 爲**單純TLB miss**或是**page fault**

(1) 如果 page在memory中➔**單純TLB miss**

單純TLB轉換失誤,把分頁load進TLB然後重試

(2) 如果 page不在memory中➔**Page fault**

發page fault exception 給 OS

跑deal with page fault流程,重點(4)

清大資工 96-04

A virtual memory system often implements a TLB to speed up the virtual-to-physical address translation. A TLB has the following characteristics. Assume each TLB entry has a valid bit, a dirty bit, a tag, and the page number. Determine the exact total number of bits to implement this TLB.

(1)It is direct-mapped (2)It has 16 entries

(3)The page size is 4 Kbytes

(4) The virtual address space is 4 Gbytes

(5)The physical memory is 1 Gbytes

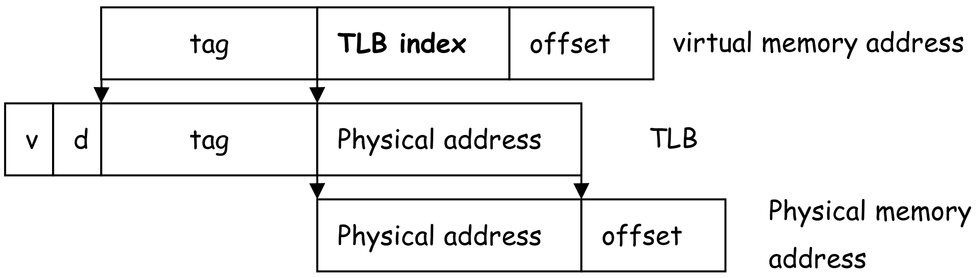
The length of virtual page number = $\log_2\left(\frac{4GB}{4KB}\right)=20 \quad bits$

The length of physical page number = $\log_2\left(\frac{1GB}{4KB}\right)=18 \quad bits$

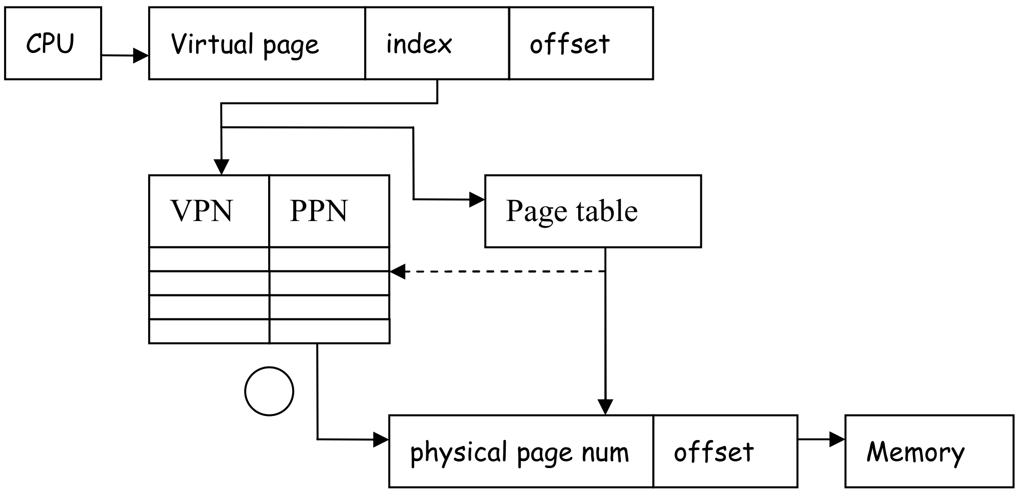
The length of index field= $\log_2 16=4$

The length of tag field= $20-4=16$

$16*(1+1+16+18)=576 \quad bits$



95 長庚資工　95 彰師資工　95 北科電通　94 淡江資工

In a virtual memory system,we usually use the TLB to make address translation fast Explain how the TLB work
用來記錄最近記憶體 address mapping 的 cache,爲了降低存取 page table 次數 TLB欄位 ➔Tag ,physical page number ,reference bit ,dirty bit  CPU Virtual page index offset VPN PPN Page table physical page num offset Memory

95 台科電機

Virtual address are 32 bits wide　　Physical address are 31 bits wide Page table size is 2KB　TLB contains 16 entry　　TLB is direct map (a) the width of each entry of TLB?　(b)TLB size?
$virtual: 32-11-4=17$ $physical: 31-11=20$ $(a) 1+20+17=38 bits$ $(b) 38*16=608 bits$

94 清大資工

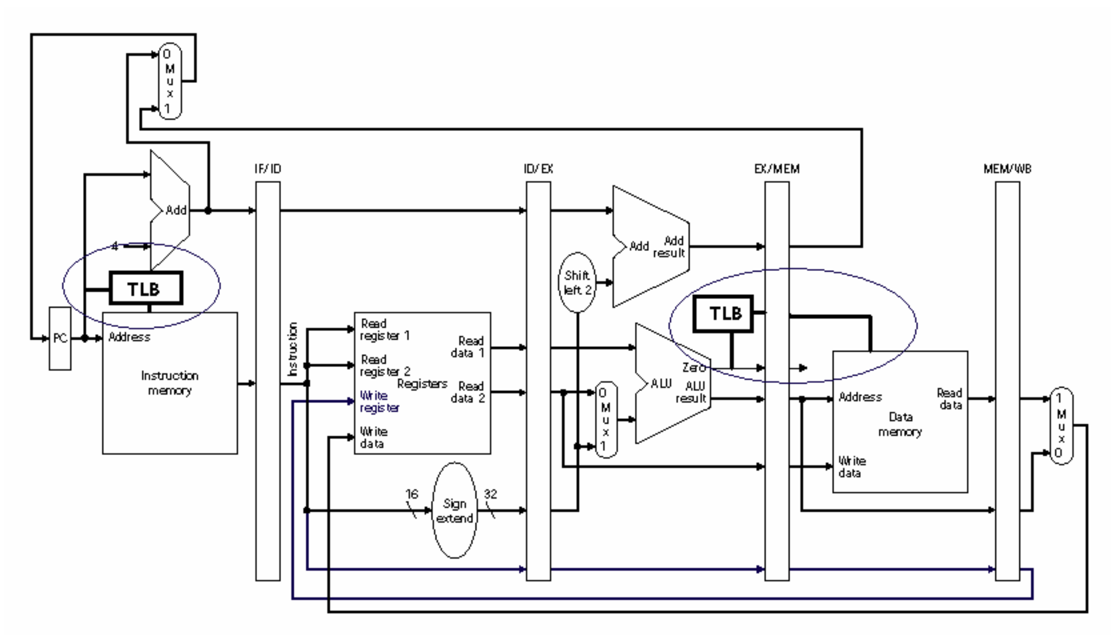
Determine the number of bits required in each entry of a TLB that has the following characteristics:The TLB is directed-mapped,The TLB has 32 entries,The page size is 1024 bytes ,Virtual byte addresses are 32 bits wide, Physical byte addresses are 21 bits wide,Note that you only need to consider the following items for each entry: The valid bit,The tag,The physical page number
the physical page number has 31 – 10 = 21 bits and TLB has 32 entries➔ index has 5 bits, then tag size = 32 – 5 – 10 = 17 So, the number of bits in each entry = 1 + 17 + 21 = 39 bits

(28) TLB in MIPS pipeline 97清大資工

Inst Fetch		Dcd/ Reg	ALU / E.A		Memory	Write Reg
TLB	I-Cache	RF	Operation		▨ WB	
			E.A.	TLB	D-Cache	

Instruction fetch 的前半階段

Execution 的後半階段



(29) page table size

Page table size= $2^{virtual} * (entry \quad width)$

高應電機 95

32-bit virtual byte address , 4KB pages , 30-bit physical byte address, extra bit(valid bit , …)➔4 bit ,what the page table size ?
$\frac{2^{32}}{4K} = 2^{20}, \frac{2^{30}}{4K} = 2^{18} \Rightarrow 2^{20} * (18+4) = 2^{20} * 3Byte = 3MB$

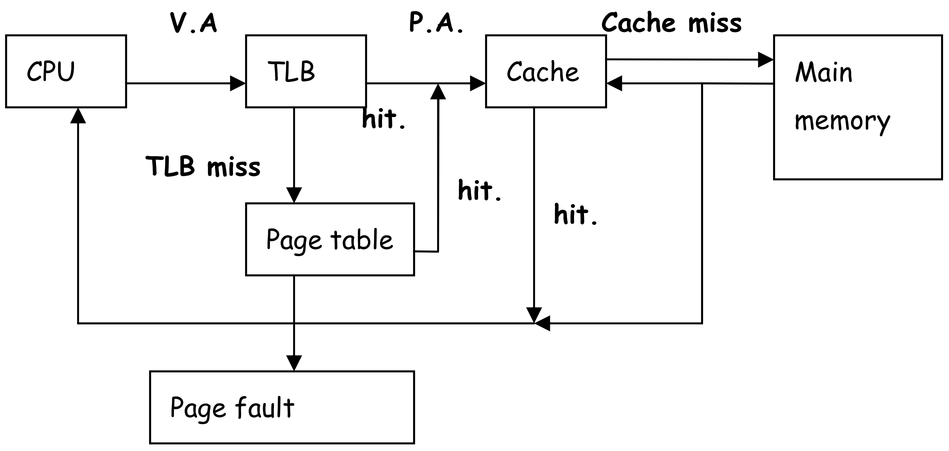
清大通訊 93

40-bit virtual byte address , 16KB pages , 36-bit physical byte address, extra bit(valid bit , …)➔4 bit ,what the page table size ?
$\frac{2^{40}}{16K} = 2^{26}, \frac{2^{36}}{4K} = 2^{22}$ $22+4 = 26bits, align = 4Byte \Rightarrow 2^{26} * 4Byte = 256MB$

(30)The possible combinations of events in the TLB, virtual memory system, and cache.
(97 交大資訊聯招)(96 中央資工) (93 清大資工) (96 長庚資工) (94 台科電子)

TLB	Page table	Cache	Possible? If so, under what circumstance?
hit	hit	**miss**	可能, TLB hit ,雖然未真正查詢分頁表
miss	hit	hit	TLB misses 後在分頁表找到資料,重試後,到 cache 找到資料
miss	hit	**miss**	TLB misses 後在分頁表找到資料,重試後,到 cache 找資料 miss
miss	**miss**	**miss**	TLB misses 後 page fault,重試後, 到 cache 找資料一定 miss
hit	**miss**	**miss**	不可能, 分頁不在記憶體中,TLB 不可能 hit
hit	**miss**	hit	不可能, 分頁不在記憶體中,TLB 不可能 hit ,資料也不會出現在 cache
miss	**miss**	hit	不可能, 分頁不在記憶體中,資料也不會出現在 cache

96中央資工

What are TLB and page table? Please describe clearly and systematically how a memory access is completed by the processor cache/main memory/TLB/page table/hard disk
 V.A P.A. Cache miss CPU TLB Cache Main memory hit. TLB miss hit. hit. Page table Page fault

(31) interleaved memory organization

(96 輔大電子)

Consider a memory hierarchy using one of the following three organizations for main memory: (a) one-word-wide memory organization, (b) wide memory organization, and (c) interleaved memory organization.
Assume that the cache block size is 4 words, that the width of organization (b) is two words, and that the number of banks in organization (c) is four. If the main memory latency for a new access is 15 memory bus clock cycles and the transfer time is 1 memory bus clock cycle. Assume that it takes 1 clock cycle to send the address to the main memory, what are the miss penalties for each of these organizations?

(a) one-word-wide memory organization:
the miss penalty would be 1 + 4 × 15 + 4 × 1 = 65 clock cycles   (4 = block size/ 1 word)
(b) wide memory organization (four-word-wide):
the miss penalty would be 1 + 2 × 15 + 2 × 1 = 33 clock cycles.   (2 = block size/ 2 word)
(c) interleaved memory organization:
the miss penalty would be 1 + 1 × 15 + 4 × 1 = 20 clock cycles.
(1 = block size/ 4 word)   (4=block size)

3th (96 中山資工)

Consider a memory hierarchy using one of the following three organizations for main memory: (a) one-word-wide memory organization, (b) wide memory organization, and (c) interleaved memory organization. Assume that the cache block size is 16 words, that the width of organization (b) of the figure is four words, and that the number of banks in organization (c) is four. If the main memory latency for a new access is 10 memory bus clock cycles and the transfer time is 1 memory bus clock cycle, what are the miss penalties for each of these organizations?

Assume that to send the address to the main memory takes 1 clock cycle.
Configuration (a): Miss penalty = 1 + 16 × 10 + 16 × 1 = 177 clock cycles.
Configuration (b): Miss penalty = 1 + 4 × 10 + 4 × 1 = 45 clock cycles. (16/4=4)
Configuration (c): Miss penalty = 1 + 4 × 10 + 16 × 1 = 57 clock cycles

1,10,1 擺完
(a)都是乘 block size=16
(b)都是乘 block size/4 word=4
(c)前面乘 block size/4 word=4 ,後面乘block size=16

(32) memory hierarchy

95 交大電子

Memory system:

(1) What is the main objective of the memory hierarchy?

(2) What is the fundamental principle that makes the memory hierarchy work? Describe the principle briefly.

(3) Briefly describe the three common strategies for each block replacement.

(4) Compare the strategies mentioned in (3) in terms of the cache miss rate and the hardware implementation cost

(1) 提供使用者能用最便宜的技術來擁有足夠的記憶體，並且利用最快的記憶體來提供最快的存取速度。

(2) 區域性原則**(Principle of locality)**：程式在任何一個時間點只會存取一小部份的位址空間稱為區域性。區域性可分為兩種不同的類型：

時間區域性**(temporal locality)**：如果一個項目被存取到，那麼它很快的會被再存取到。

空間區域性**(spatial locality)**：如果一個項目被存取到，那麼它位址附近的項目也會很快被存取到。

(3)

direct map:一種快取架構,每一個 memory location 只對應到一個 cache location

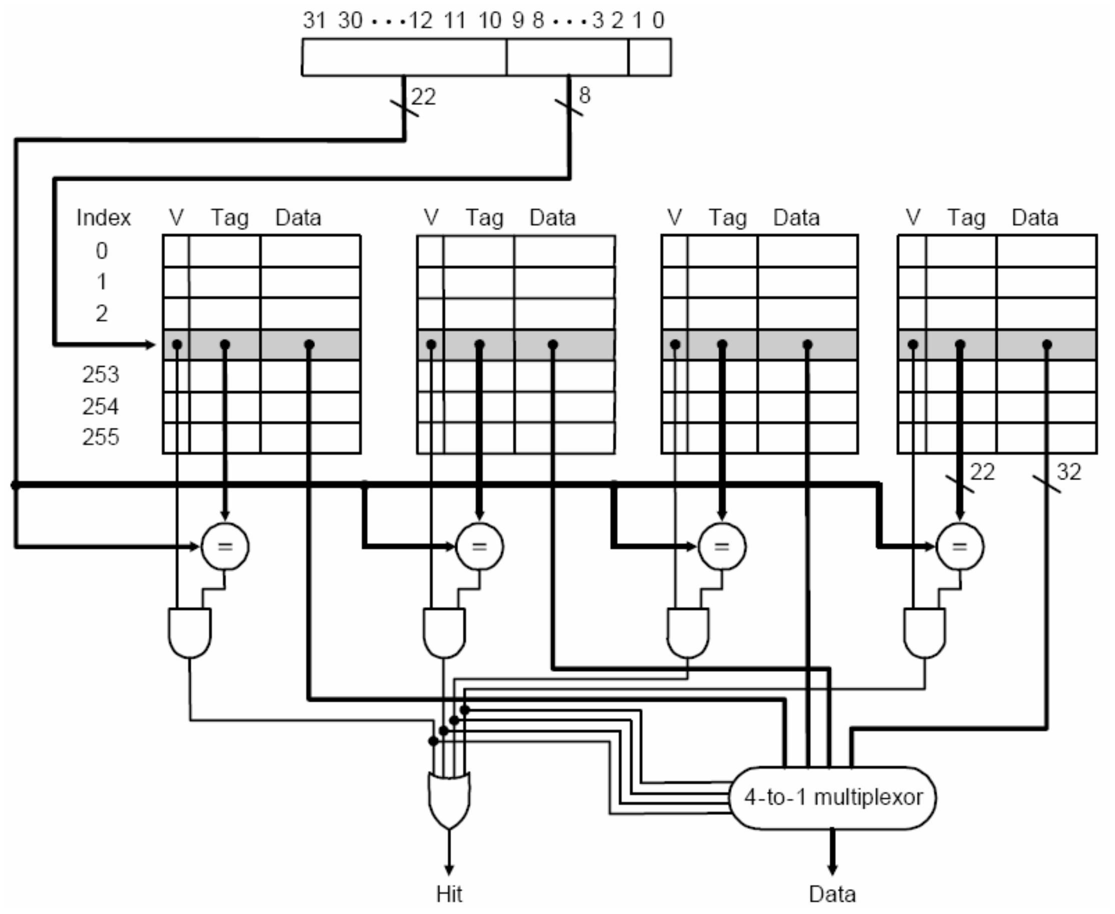
Set-associative cache:一種快取架構,每一個 memory location對應到固定數量(unless cache location

Fully associative cache:一種快取架構,每一個 memory location能對應到cache中的每一個location的

(4)

strategies	miss rate	Cost
Direct-mapped	High	Low
Set-associative	Medium	Medium
Fully associative	Low	High

(34) 4-Way Set Associative Implementation



(35) Write-through and Write-back(94中山資工)(96輔大資工) (96彰師資工) (94台大電機) (92台大資工) (95清大資工)(94中央資工)(92台大資工) (95清大資工考pros) (96雲科資工)(95東華資工) (94中山電機) (92暨南資工) (90中山資工)

	Write-through	**Write-back**
定義	快取的一種寫入機制 ,當有資料寫快取時,會同時更新快取和記憶體資料,用來確定資料一制性	快取的一種寫入機制 ,平常寫入料時,只會更新快取,當block要被代時,才寫回(更新) 下層的memory
pros	(a)不會有 cache inconsistent的問題 (b)易於實作	(a)多次寫入,只需寫回memory一次,速度較快，校能較高 (b) 當block寫回memory,system有更大的頻寬可用
cons	寫入記憶體時間長,速度較慢 校能較低	實作困難 會有 cache coherence的問題

(36)TLB miss and page fault

(a)How to deal with page fault? (95高應電機)

(1) process發一個page fault trap給OS,OS取得控制權

(2) OS由internal table(page table)判斷此reference是否valid

Invalid➔abort process

Valid➔page 在disk中

(3)在 memory 中找一個 free(empty) frame

(4)把 需要的page從disk讀入memory中

(5)把page table中剛讀如的entry中valid bit update成1

(6)重新執行剛才發生trap的指令

(b)降低 page table size 的5種方法 (97台大電機) (95元智資工)

(1) 使用 limit register 限制每個process的page table size

(2) 使用2組獨立的 page tables 和 limits register

(3) 使用反轉分頁表 (inverted page table)

(4) 使用多層分頁表 (Multiple levels page table)

(5) Page tables to be paged

(c)Handling TLB Miss? (95 長庚資工) (95 北科電通)

如果 TLB miss,先判斷 爲單純TLB miss或是page fault

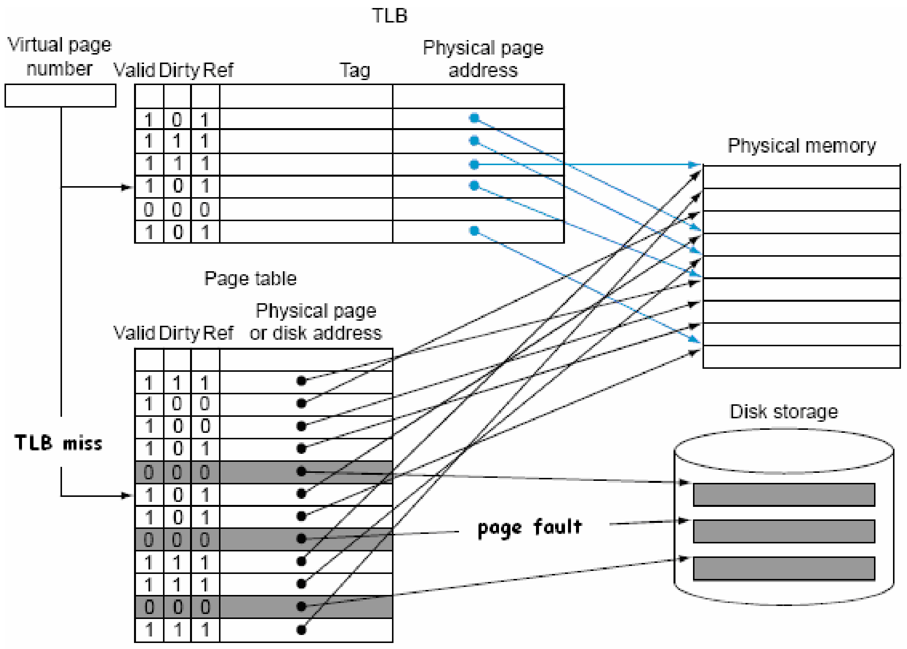
(1) 如果 page在memory中➔單純TLB miss

單純TLB轉換失誤,把分頁load進TLB然後重試

(2) 如果 page不在memory中➔Page fault

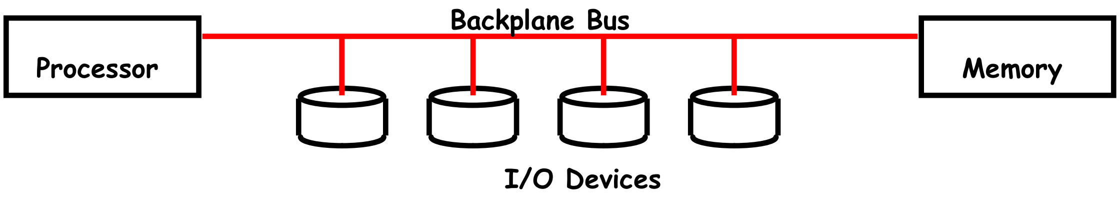
發page fault exception 給 OS

跑deal with page fault流程,重點(a)

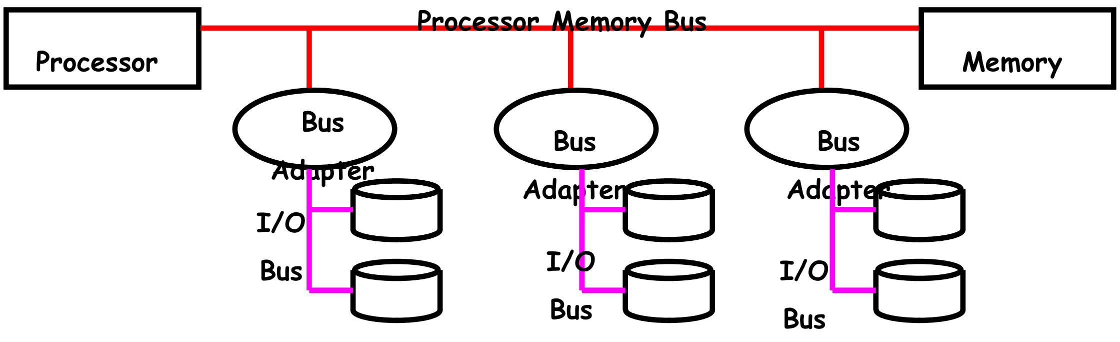


(37)Bus System (90台大資工)

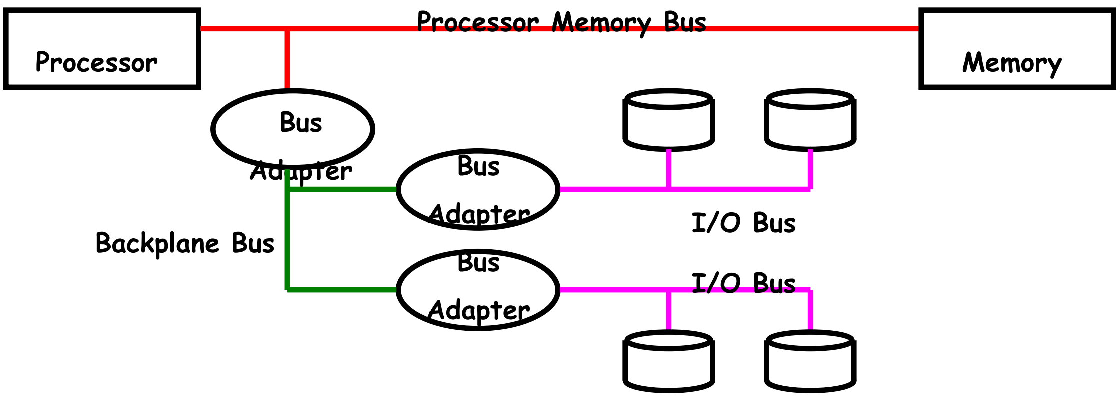
(1)Computer System with One Bus: Backplane Bus



(2) A Two-Bus System



(3)A Three-Bus System



(38) Synchronous/Asynchronous Bus (92 台大資工)

	Synchronous Bus	Asynchronous Bus
	控制線中包含 clock,使用一個與 clock 有關的通訊協定	控制線不須包含 clocked,使用 handshaking 通訊協定
優點	(1)非常少的硬體邏輯電路,速度快 (2)沒有 handshaking 協定的負擔	(1)可容納廣泛範圍的 devices (2)不需考慮 clock skew 的問題,bus 可以很長
缺點	(1)Bus 上的每個 device 必須跑相同的 clock rate (2)要考慮 clock skew 的問題,bus 線不能太長	(1)速度慢 (2)需要 handshaking 協定與額外控制電路
常用	常用於 cpu 與 memory 間的 bus	I/O bus

(39)Polling I/O,Interrupt driven I/O ,DMA (94台大資工) (95朝陽資工: DMA) (94中央通訊(1)&(2)) (94政大資科(1)&(3)) (94淡江電機(2)&(3)) (94淡江資工 3) (92 海大電機) (92 台師資工 prons) (91 成大電機(1)&(2))(91 中正電機) (91 台科電子 3) (90 交大資工) (90 中央電機(2)&(3))

	Polling I/O (1)	Interrupt driven I/O(2)	DMA(3)
Def	I/O device 只要將 information 放進 status register，CPU 會周期性的檢查並取得 information 來得知需要服務的 device	I/O device 當需要與完成 I/O 時發 interrupt 給 CPU,因為 I/O interrupt 是 asynchronous I/O,所以當 I/O 做傳輸時,CPU 可做其他的 operation	提供一個 device controller，讓 I/O device 能夠直接在記憶體做資料的傳輸，不需要 CPU 的參與
prons	簡單容易執行，可利用軟體來更改 CPU polling 的順序	處理器只需處理在 interrupt 時處理 I/O (I/O request,I/O complete) 不用浪費許多時間在 polling 上面	資料由 I/O 與 Memory 相互傳輸, 不需 cpu 介入, 適用於高速裝置
cons	浪費太多的處理器時間在 polling 上	大量的 hardisk 讀寫仍然太慢	需要額外hardware 如DMAC

(40)Bus arbitration scheme(95清大電機)(95高雄電機) (92雲科電子)

Daisy chain arbitration (92 中央資工)

每個 device 都有一個 priority, arbitrator 由 priority 高到低尋視,有最高 priority 又提出 request 將得到匯流排使用權

Centralized, parallel arbitration

每個 device 都有自己的 request line, Centralized arbitrator 會從發出 request 的 device 中選一個

Distributed arbitration by self-selection

每個 device 都有自己的 request line, 發出 request 的 device 的 device 自己決定匯流排使用權給誰

Distributed arbitration by collision detection

每個設備都可以對匯流排發出請求，如果匯流排偵測到碰撞，就重新發出請求

3th　95 台大資工,94 交大資工,94 輔大電子

Consider the following computer system: *A CPU that sustains 3 billion instructions per second and averages 100,000 instructions in the operating system per I/O operation *A memory backplane bus capable of sustaining a transfer rate of 1000 MB/sec *SCSI Ultra320 controllers with a transfer rate of 320 MB/sec and accommodating up to 7 disks *Disk drives with a read/write bandwidth of 75 MB/sec and an average seek plus rotational latency of 6 ms.If the workload consists of 64 KB reads (where the block is sequential on a track) and the user program needs 200,000 instructions per I/O operation, find the maximum sustainable I/O rate and the number of disks and SCSI controllers required. Assume that the reads can always be done on an idle disk if one exists (i.e., ignore disk conflicts).
Maximum I/O rate for CPU=(Instruction execute rate)/(Instruction per I/O) $=\frac{3*10^9}{200*10^3+100*10^3}=10000$ (I/O)/second **Maximum I/O rate for Bus=**(Bus bandwidth)/(Byte per I/O) $=\frac{1000*10^6}{64*10^3}=15625$ (I/O)/second **Time per I/O at disk**=Seek + Rotation + Transfer $=6ms+\frac{64K}{75MB/\sec}=6.9ms$ **Transfer rate** =(transfer size)/ (transfer stime)= $\frac{64K}{6.9ms}=9.56MB/\sec$

(95元智資工) (95輔大資工)

Suppose we have a benchmark that executes in 100 seconds of elapsed time, where 90seconds is CPU time and the rest is I/O time. If the CPU time improves by 50% per year for the next five years but I/O time does not improve, how much faster will our program run at the end of the five years?

Elapsed Time = CPU time + I/O time

After n years	CPU time	I/O time	Elapsed Time	% I/O time
0	90	10	100	10%
1	90/1.5=60	10	70	14%
2	60/1.5=40	10	50	20%
3	4/1.5=27	10	37	27%
4	27/1.5=18	10	28	36%
5	18/1.5=12	10	22	45%

Over five years:

CPU improvement = 90/12 = 7.5 **BUT** System improvement = 100/22 = 4.5

2th (93 中央電機)

Consider the following computer system: *A CPU that sustains 3 billion instructions per second and averages 50,000 instructions in the operating system per I/O operation *A memory backplane bus capable of sustaining a transfer rate of 100 MB/sec *SCSI Ultra320 controllers with a transfer rate of 20 MB/sec and accommodating up to 7 disks *Disk drives with a read/write bandwidth of 5 MB/sec and an average seek plus rotational latency of 10 ms.If the workload consists of 64 KB reads (where the block is sequential on a track) and the user program needs 100,000 instructions per I/O operation, find the maximum sustainable I/O rate and the number of disks and SCSI controllers required. Assume that the reads can always be done on an idle disk if one exists (i.e., ignore disk conflicts).
Maximum I/O rate for CPU=(Instruction execute rate)/(Instruction per I/O) $=\frac{3*10^9}{50*10^3+100*10^3}=2000$ (I/O)/second **Maximum I/O rate for Bus**=(Bus bandwidth)/(Byte per I/O) $=\frac{1000*10^6}{64*10^3}=15625$ (I/O)/second **Time per I/O at disk**=Seek time + Rotation time + Transfer time $=10ms+\frac{64K}{5MB/\sec}=22.8ms$ **Transfer rate** =(transfer size)/ (transfer stime)= $\frac{64K}{22.8ms}=2.74MB/\sec$

Check youself

Which of the following are true about RAID levels 1, 3, 4, 5, and 6? 1. RAID systems rely on redundancy to achieve high availability. 2. RAID 1 (mirroring) has the highest check disk overhead. 3. For small writes, RAID 3 (bit-interleaved parity) has the worst throughput. 4. For large writes, RAID 3, 4, and 5 have the same throughput.
1, 2, 3 and 4

2th

What is the average time to read or write a 512-byte sector for a typical disk rotating at 5400 RPM? The advertised average seek time is 12 ms, the transfer rate is 5 MB/sec, and the controller overhead is 2 ms. Assume that the disk is idle so that there is no waiting time.
Disk Access Time = Seek time + Rotational Latency + Transfer time + Controller Time + Queuing Delay = 12.0 ms + 0.5 / 5400 RPM + 0.5 KB / 5 MB/s + 2 ms = 12.0 ms + 5.6 + 0.1 + 2 = 19.7 ms If the measured average seek time is 25% of the advertised average time, the answer is 3 ms +5.6 ms + 0.1 ms + 2 ms = 10.7 ms

3th

What is the average time to read or write a 512-byte sector for a typical disk rotating at 10,000 RPM? The advertised average seek time is 6 ms, the transfer rate is 50 MB/sec, and the controller overhead is 0.2 ms. Assume that the disk is idle so that there is no waiting time.
Disk Access Time = Seek time + Rotational Latency + Transfer time + Controller Time + Queuing Delay = 6.0 ms + 0.5 / 10,000 RPM + 0.5 KB / 50 MB/s + 0.2 ms = 6.0 ms + 3.0 + 0.01 + 0.2 = 9.2 ms If the measured average seek time is 25% of the advertised average time, the answer is 1.5 ms + 3.0 ms + 0.01 ms + 0.2 ms = 4.7 ms

92 政大資科

Explain the Flynn`s taxonomy of parallel computer
SISD: single instruction stream, single data stream 單一 processor decode 單一 instruction , 如傳統計算機 SIMD: single instruction stream, multiple data stream 單一 instruction 同時控制數個 處理器,每個 processing unit 有自己的 memory, 例:vector processor ,array processor MISD: multiple instruction stream, single data stream 一串 data 由不同的 processor 執行,每個 processor 執行不同的運算,從未實作過 MIMD: multiple instruction stream, multiple data stream 數個 processor 同時對不同的 data set 執行不同的指令,如 multiprocessor 或 Multi-computer system

(41)RAID (Redundant arrays of Inexpensive disk) (95暨南資工) (94輔大資工 0,1)

RAID	Name $\frac{used}{Total}disk$	
0	stripe (no redundancy) $\frac{N}{N}$	將邏輯上的連續區塊分配到不同硬碟的配置方式 **沒有redundant硬碟**(不fault-tolerant) 在所有 disks 間平行寫入平行讀取 **I/O 速度最快**
1	Mirroring $\frac{N}{2N}$	**可靠性最高(reliablity),容錯(fault tolerate) ,易recovery** 當寫入時會同時寫到redundant硬碟,**最浪費硬碟** 資料有錯時只需到mirror中讀取
3	Bit-interleaved parity $\frac{N}{N+1}$	bit-level striping, 使用額外一部同位磁碟機 (parity disk) **因讀寫橫跨所有disk,只支援單一讀寫** 對於 RAID1花較多的時間recovery,花較少的時間
4	block-interleaved parity $\frac{N}{N+1}$	相似 RAID 3,但使用 block-level striping 對於 RAID 0, 只多了 parity block, **支援平行讀取** 缺點:每次寫入,同位元必需不斷更新,同位 disk 寫入爲瓶頸
5	Distribute block-interleaved parity $\frac{N}{N+1}$	改善 RAID4 相異同位元爲於同 1 個 disk 上,RAID 5 把 parity bit 分散在全部的 disk,所以**支援平行讀取寫入**
6	P+Q redundancy $\frac{N}{N+2}$	同一陣列中容許兩個硬碟同時失效 計算 2 組 parity information 來提高更高的容錯 (fault tolerance),RAID5 的 2 倍 **highest availability ,lowest probability of data loss**
0+1 (10)	Mirror across stripes $\frac{N}{2N}$	先 mirror 再 stripe 同時具有資料容錯與讀寫速度快的特色 **Performance higher than 1+0** has the same **use ratio** as RAID 0 has the same **reliability** as RAID 1 Very expensive / High overhead
1+0 (01)	Stripe across mirrors $\frac{N}{2N}$	先 stripe 再 mirror **Reliability higher than 0+1**

RAID 1 and RAID 5 廣泛的運用於 servers (about 80%)

95 清大資工

Consider a loop branch that branches nine times in a row, then is not taken once. Assume that we are using a dynamic branch prediction scheme. (a) What is prediction accuracy for this branch if a simple 1-bit prediction scheme is used (b) What is prediction accuracy for this branch if a 2-bit prediction scheme is used?
(a) 80%，因為第一次和最後一次會猜錯因此其正確率為(8/10) × 100% = 80% (b) 90%，最後一次會猜錯因此其正確率為(9/10) × 100% = 90%

95 清大資工

The majority of processor caches today are direct-mapped, two-way set associative, or four-way set associative, but not fully associative. Why?
Because the costs of extra comparators and delay imposed by having to do the compare for fully associative are too high

95 清大資工

What feature of a write-through cache makes it more desirable than a write-back cache in a multiprocessor system (with a shared memory)? On the other hand, what feature of a write-back makes it more desirable than a write-through cache in the same system?
(a) Write-through improve the coherence between share memory and cache, thereby reduces the complexity of the cache coherence protocol. (b) Write-back reduces bus traffic and thereby allows more processors on a single bus.

Compare Micro-programming control and Hardware control(94 中山資工) (95 中原資工)

	Hardware control	**Micro-programming**
	又稱有限狀態機法(Finite State Machine: FSM)。 利用 FSM 將指令執行之過程分成數個狀態，並分析每個狀態要產生的控制訊號，決定下一個時脈週期要轉換到哪一個狀態。最後用硬體來製作控制電路	利用微指令(microinstruction)，來定義每一個狀態的控制訊號，執行一個微指令就會送出此指令所描述的控制訊號
prons	速度快 硬體成本便宜	修改設計容易 能處理較複雜的指令集
cons	修改設計不容易 只適合處理較簡單的指令集	速度慢 只適合處理較簡單的指令集 硬體成本較高

95 成大資工　96 清大電機

The following program tries to copy words from the address in register $a0 to the address in register $a1 and count the number of words copied in register $v0. The program stops copying when it finds a word equal to 0. You do not have to preserve the contents of registers $v1, $a0, and $a1. This terminating word should be copied but not counted.

```
Loop:    lw $v1, 0($a0)
         addi $v0, $v0, 1
         sw $v1, 0($a0)
         addi $a0, $a0, 1
         addi $a1, $a1, 1
         bne $v1, $zero, loop
```

There are multiple bugs in this MIPS program. Please fix them and turn in bug-free

```
         addi $v0, $zero, -1
Loop:    lw $v1, 0($a0)
         addi $v0, $v0, 1
         sw $v1, 0($a1)
         addi $a0, $a0, 4
         addi $a1, $a1, 4
         bne $v1, $zero, Loop
```

96 中央電機

While executing the MIPS code shown below, what is the target address of the branch instruction if it is taken? (starting address of code segment is 28dec.)

```
    lw $4, 50($7)
    beq $1, $4, 3
    add $5, $3, $4
    sub $6, $4, $3
    or $7, $5, $2
    slt $8, $5, $6
```

Beq 在 address 32,如果 taken 32+4*3+4=48　　多加 4 爲算盤中的定義

94 中正資工,90 交大資工

MIPS ISA Design principle
simplicity favors regularity(單一大小指令,三個 register 運算元, 位址固定) smaller is faster(只有 32 個 register) good design demands compromise(所有指令相同長度,都可以提供夠大的位址及常數値) make the common case fast(PC-relative , immediate addressing modes 特別加快速度)

96 中山資工

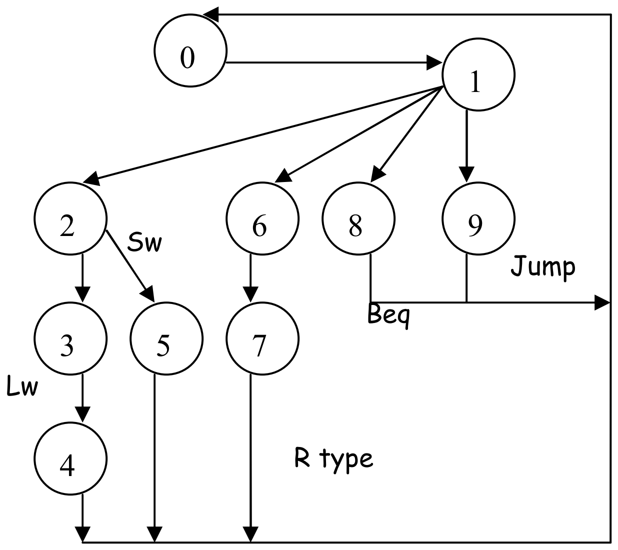
Consider the following machines, and compare their performance using the following instruction frequencies: 25% Loads, 13% Stores, 47% R-type instructions, and 15% Branch/Jump.

M1: The multicycle datapath shown in Fig. 1 with a 4 GHz clock.

M2: A machine like the multicycle datapath of Fig. 1, except that register updates are done in the same clock cycle as a memory read or ALU operation. Thus in Fig. 2 (which shows the complete finite state machine control of Fig. 1), states 6 and 7 and states 3 and 4 are combined. This machine has a 3.2 GHz clock, since the register update increases the length of the critical path.

M3: A machine like M2 except that effective address calculations are done in the same clock cycle as a memory access. Thus states 2, 3, and 4 can be combined, as can 2 and 5, as well as 6 and 7. This machine has a 2.8 GHz clock because of the long cycle created by combining address calculation and memory access.

Find the effective CPI and MIPS (million instructions per second) for all Machines



	Frequency	M1	M2	M3
Load CPI	25%	5	4	3
Store CPI	13%	4	4	3
R-type CPI	47%	4	3	3
Branch/jump CPI	15%	3	3	3
Effective CPI		4.1	3.38	3
MIPS		976	946	933

Compare the performance for single-cycle, multicycle, and pipelined control by the average instruction time using the following instruction frequencies (25% loads, 10% stores, 11 % branches, 2% jumps, and 52% ALU instructions) and functional unit times (200 ps for memory access, 100 ps for ALL) operation, and 50 ps for register file read or write). For the multicycle design, the number of clock cycles for each instruction class is shown in Fig. 2. For the pipelined design, loads take 1 clock cycle when there is no load-use dependence and 2 when there is. Branches take 1 when predicted correctly and 2 when not. Jumps always pay 1 full clock cycle of delay, so their average time is 2 clock cycles. Other instructions take 1 clock cycle. For pipelined execution, assume that half of the load instructions are immediately followed by an instruction that uses the result and that one-quarter of the branches are mispredicted. Ignore any other hazards.

For single-cycle machine:

CPI = 1

Clock cycle time = 200 + 50 + 100 + 200 + 50 = 600 ps

Average instruction time = 1 × 600 = 600 ps

For multicycle machine:

CPI = 0.25 × 5 + 0.1 × 4 + 0.11 × 3 + 0.02 × 3 + 0.52 × 4 = 4.12

Clock cycle time = 200 ps

Average instruction time = 4.12 × 200 = 824 ps

For pipeline machine:

Effective CPI = 1.5 × 0.25 + 1 × 0.1 + 1 × 0.52 + 1.25 × 0.11 + 2 × 0.2 = 1.17

Clock cycle time = 200 ps

Average instruction time = 1.17 × 200 = 234 ps

The relative performance of the three machines is

Pipeline > single-cycle > multi-cycle

in terms of average instruction time.

(a) The frequency of all loads and stores in gcc is 36%. Assume an instruction cache miss rate for gcc of 2% and a data cache miss rate of 3%. If a machine has a CPI (clock cycles per instruction) of 2 without any memory stalls and the miss penalty is 50 cycles for all misses, determine how much faster a machine would run with a perfect cache that never missed. (b) Suppose we double the clock rate of the machine, how much faster will the machine be with this faster clock, assuming the same miss rate as describe above? (c) double the processor rate (d)get new Memory is twice as faster
(a) 令指令數爲 I Instruction miss cycle=I*2%*50=I Data miss cycle=I*3%*36%*50=0.54I Total miss cycle = I+0.54I=1.54 I Perfect cache 快了 (2+1.54)/2=1.77 (b) clock rate 快了 2 倍➔Miss penality 快了 2 倍 **50 cycles➔100 cycles** Instruction miss cycle=I*2%*100=2I Data miss cycle=I*3%*36%*100=1.08I Total miss cycle = 2I+1.08I=3.08 I 上例快了 $\frac{3.54}{5.08*\frac{1}{2}}=1.39$ (c) processor 快了 2 倍➔ CPI 由 2 變爲 1 Perfect cache 快了 (1+1.54)/1=2.54 (d)Memory 快了 2 倍➔Miss penality 變 0.5 倍 **50 cycles➔25 cycles** Instruction miss cycle=I*2%*25=0.5I Data miss cycle=I*3%*36%*25=0.27I Total miss cycle = 0.5I+0.27I=0.77 I Perfect cache 快了 (2+0.77)/2=1.385

(42)Key Word

CPI (91 暨南資工)(90 清大電機)

CPI 為指令時脈週期數(clock cycle per instruction)的縮寫

表示平均每個指令所需花費的時脈週期數

Throughput

在某一時間區間內所完成的工作量

Arithmetic mean **Geometric mean**

與總執行時間成正比的平均執行時間 $\frac{\sum_{i=1}^{n} Time_i}{n}$ $\sqrt[n]{\prod_{i=1}^{n} Time_i}$

Amdahl`s Law(96 朝陽資工) (96 彰師資工) (96 北科電通) (96 暨南資工) (95 朝陽資工)

一個系統中總效能的改善,會受限於改善部份所佔有系統的比例

Spilling register

把較不常使用的變數放入記憶體的程序

big-endian(MIPS) (94 政大資料) (92 長庚資工)

記憶體放置方法,將 least significant byte 放在**較低**記憶體位址

little-endian (94 政大資料) (92 長庚資工)

記憶體放置方法,將 least significant byte 放在**較高**記憶體位址

Procedure frame or activation record or stack frame

(96交大資聯) (92暨南資工) (92 長庚資工)(92大同資工)

堆疊的某個區段, 儲存程序(procedure)的**暫存器與區域變數**

frame pointer (96 交大資聯)

MIPS 用來 pointer 到 **Procedure frame** 第一個 word 的 pointer

Loop unrolling

為了降低 loop 的次數(overhead),複製程式主體,實現 compile optimize 的一種方法

Biased notation

用 111….111 表示最大的正數, 用 000….000 表示最小的負數

用 100….000 表示零,**因此所有數加上 Bias 都會大於等於 0**

overflow(floating point) (96 大同資工)

正的指數大到指數欄位無法表示的情況(正數大到硬體無法表示的情況)

Underflow(floating point)

負的指數大到指數欄位無法表示的情況(正數小到硬體無法表示的情況)

Guard bit

IEEE 浮點數計算過程中,右邊 2 個額外位元的第 1 個位元,用來提高捨去的精準度

Round bit

IEEE 浮點數計算過程中,右邊 2 個額外位元的第 2 個位元,用來提高捨去的精準度

sticky bit

用來提高捨去的精準度的第 3 個 bit,當捨入時,round bit 右邊有非 0 的 bit,則 set

units in the last place (ulp) (95元智資工)

the number of bits in error in the least significant bits of the significand between the actual number and the number that can prepresented

e.g. $2.3656 \times 10^2 = 2.37 \times 10^2$ ← 0.44 ulp (unit in the last place)

sign extension(96 中央通訊)

增加一個資料的長度,只增加高位元,增加的高位元由原本的 sign bit 複製而得

Gradual Underflow

允許一個有效數字在精確度上逐漸遞減，一直到變成 0 爲止

branch taken

branch 條件成立且新的PC位址爲條件跳躍目標位址(branch target address)

branch not taken

branch 條件不成立,PC位址爲下一個執行位址

delayed branch(92大同資工) (91台科電子)

不論條件跳躍是否成立,緊接在條件跳躍指令後的那個指令一定會被執行

Pipeline: (90 中央電機

多個指令重疊執行 (Multiple instructions are overlapped in execution)

Hazard

下一個 instruction 無法在下一個 clock cycle 順利執行

Forwarding(96 大同資工) (94 中興電機) (92 台師資工)

亦稱爲 bypassing,一種解決 data hazard 的方法,提早從內部 buffer 捷取缺少的資料,而非等到這些資料到達程式設計者可見的 register 或 memory

Load-use data hazard

一種特定的 data hazard 形式,當需要 load 指令所 load 出來的資料無法立即獲得

MIPS指令中常發生於一個load指令緊接一個R-type指令

Pipeline stall

亦稱爲 bubble . stall 是爲了要解決hazard

Branch prediction(96中原資工) (95中原資工) (92台師資工) (92台科電機)

用來解決control hazard的方法,先假設條件跳躍的結果往下進行,而非等到結果確定

Flush(instruction)

丟棄 pipeline 中的 instruction,一般發生於未預期的事件

Dynamic Branch prediction (94 台師資工) (92 輔大資工)

在 runtime 中使用 runtime information 來預測分支

Branch delay slot(96暨南資工) (96中央通訊)(92清大電機)

在 delayed branch instruction後, 緊接一個不影響條件跳躍的指令

Imprecise interrupt

亦稱爲imprecise exception,pipeline中的interrupt,與造成interrupt或exception的指令並無關聯,即放入EPC的指令位址,並非真正發生exception的那個指令

precise interrupt (93台大電機) (MIPS有支援precise interrupt)

亦稱爲precise exception,pipeline中的interrupt或exception,總是關聯到造成interrupt或exception的指令

multiple issue (92 暨南資工)

在一個 clock cycle 中可 issue 一個以上的指令

Static multiple issue

執行前, 由 **compiler** 做指令編排

Dynamic multiple issue execution(Out-of-order execution)95 中原資工,92 暨南資工)

在一個clock可以issue多個instruction的方法,在**processor**執行指令時,由processor做指令編排，若某一指令堵住時,不會影響後面的指令

Speculation

Compiler或cpu先猜測指令的結果,爲了要移除指令執行時與其他指令的相依性

Register renaming

Register重新命名,爲了要移除antidepedence

Antidepedence (name depedence)

單純爲register 名稱重覆,而非真正資料相依性

Dynamic pipeline scheduling (92 長庚資工) (94 中山資工)

由硬體的支援把指令順序重排避免stall

Superpipeline(94中山資工) (92元智資工) (90中央電機) (90中山資工)

pipeline理想上speedup應該等同於pipeline得stage數目,所以superpipeline就是加深pipeline stage數目的pipeline架構,因stage較多,所以performance較好,但hazard overhead較大

Superscalar(96中原資工) (95中原資工) (95中央資工)(96中興資科) (92元智資工) (91台科電子) (90清大電機) (90中央電機) (90中山資工)

是一個高等pipeline的架構 ,某些functional units 會replicate多組, 使在每個clock cycle執行一個以上的指令, 將CPI降低到小於一，提昇計算速度

Superpipeline with Superscalar 的分別

Superscalar 爲一個clock cycle 平行執行 n(2)條指令

Superpipeline 爲一個指令分到 1/n (1/2)clock cycle

固都是一個clock cycle 可以issue 2條以上的指令

VLIW (94 朝陽資工) (92 長庚資工) (90 清大電機) (90 中央電機)

VLIW(Very Long Instruction Word)，**編譯器(compiler)將多個 independent 指令結合成一個長的指令字串同時執行**

Instruction-level parallelism (ILP)

指令間的平行程度,有2方法來實作,一為增加pipeline 深度,二為複製電腦內部單元(functional unit)

Out-of-order execution

Pipeline執行時,**一個instruction blocked時,不會影響後面instruction的執行**

Principle of locality(96 朝陽資工) **(**95 台科電子)

程式在一個時間點只會存取一小部份的位址空間

Temporal locality(96 朝陽資工) (95 台科電子)

如果有一 data location 被 reference 到,不久後將再被 reference

Spatial locality(96 朝陽資工) (95 台科電子) (92 清大電機)

如果有一 data location 被 reference 到,它附近的 address 將很快被 reference

Miss Penalty(96 暨南資工)

將下層記憶體(lower memory)所需的資料block,取代上層記憶體(upper memory)的資料block所花的時間

Split cache

Cache中 ,一塊為 instruction cache, 另一塊為data cache cache,hit rate 高於等容量之Unified cache

Global miss rate

多層快取中,在全部階層存取,發生失誤的比例

Local miss rate

在存取快取某一層中,所發生失誤的比例

Physical address

memory unit 所看見的 address,格式**< frame number , offset >**

Page fault

所存取的 page,不在 memory 中

Virtual address

由 CPU 所產生的邏輯位址 ,格式 **< page number , offset >**

cache inconsistent

當我們寫入資料只更新快取,而沒有更新記憶體,就會造成inconsistent的情形

Virtually addressed cache

使用 virtual address 做索引快取和標籤,為 virtual indexed,virtual tagged,不需 **TLB**

Aliasing (96清大電控) (92大同資工)

2個virtual address 對到相同的記憶體分頁

TLB(translation lookaside buffer) (96 中央資工) (93 台大電機)(95 大同資工)

Page table 的一個 cache

用來記錄最近記憶體 address mapping 的 cache,為了降低存取 page table 次數

提高 virtual address 與 physical address 的轉換效率

TLB欄位 ➔Tag ,physical page number ,reference bit ,dirty bit

Nonblocking cache

當cache還在處理快取時,還允許processor存取的cache

Virtually indexed, physically tagged

以 Virtual address 做索引,使用 physical address 當標籤

虛擬索引獲得效能的好處,及從physical addressed cache獲得架構設計簡化的好處

不會有別名的問題(virtual indexed , virtual tagged有 Aliasing的問題)

Write buffer(95清大電機) (96雲科資工)

write through時會同時更新cache and memory,為了減少CPU等待memory的寫入時間,可設計一個write buffer,資料先寫入buffer,等到有時間再從buffer寫入memory

interleaved memory(95大同資工)

memory由多個memory bank所組成,資料的存取是採用連續的資料擺在連續不同的memory bank上,可大幅提升read/write 效能

cache coherency(94 暨南資工)

多處理器的環境下,某個processor中的cache某個block被modify,**需確保memory與其他processor中block也需update**

RAID (93 台大電機) (95 朝陽資工)(95 台科電機)(94 淡江資工) (95 政大資料)

Redundant arrays of Inexpensive disk

一種用較小並較便宜的硬碟組成陣列的磁碟架構,以增加performance和reliablity

Bus transaction

一連串的匯流排操作,包含一個request or respons, ,有可能會傳送資料

Handshaking protocol

一種用於非同步匯流排傳輸,確定sender 和receiver都完成動作,才進行下一個步驟

Split transaction protocol (93 交大資工**)**

增加匯流排頻寬的方法,當device不使用匯流排時,就把匯流排控制權釋放掉

Memory-mapped I/O

一種I/O技術,assign一塊memory給I/O用,對這些位址下指令,等於對I/O下指令

Striping

將邏輯上的連續區塊分配到不同硬碟的配置方式

Mirroring

將完全相同的資料寫入多個硬碟,用來增加資料的 availiablity

Bus Arbitration

決定匯流排上哪一個device下次可以使用匯流排的方式

Snoopy Protocol (find)

在多重處理器內，維持所有快取控制器之間的快取一致性