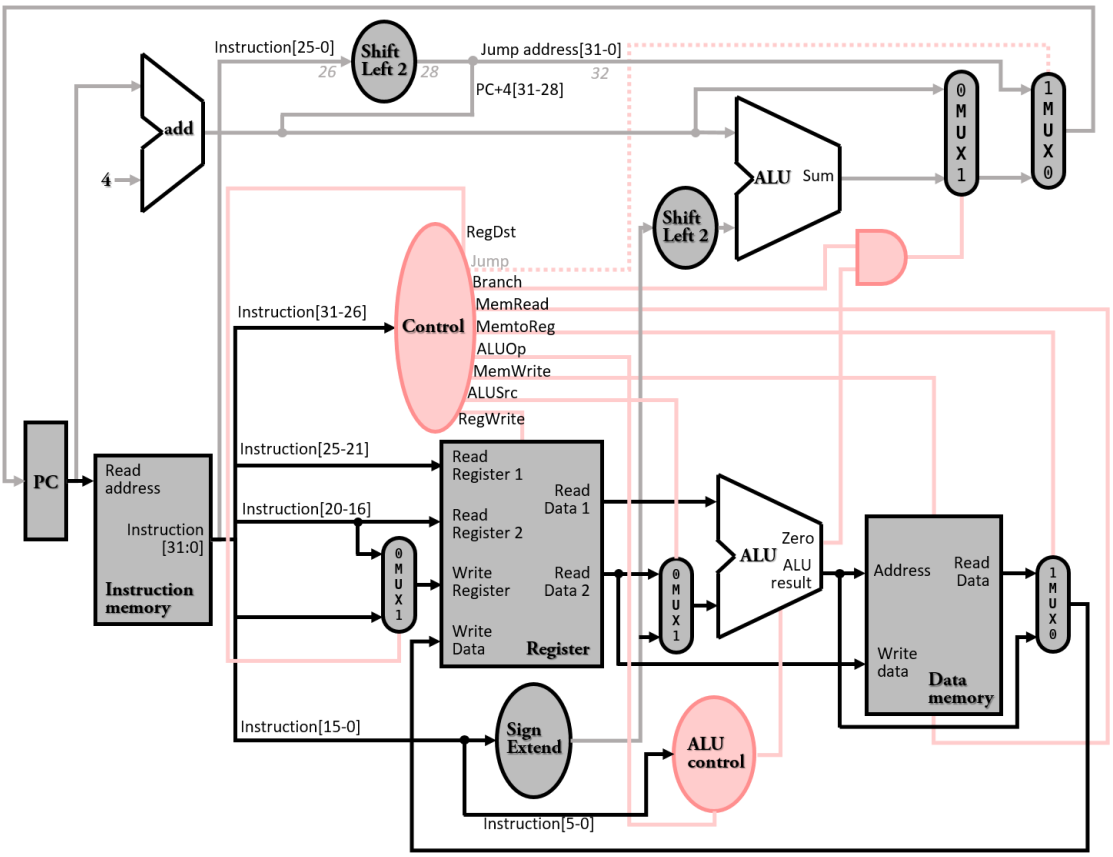
CH4、Datapath

處理器的資料路徑與控制



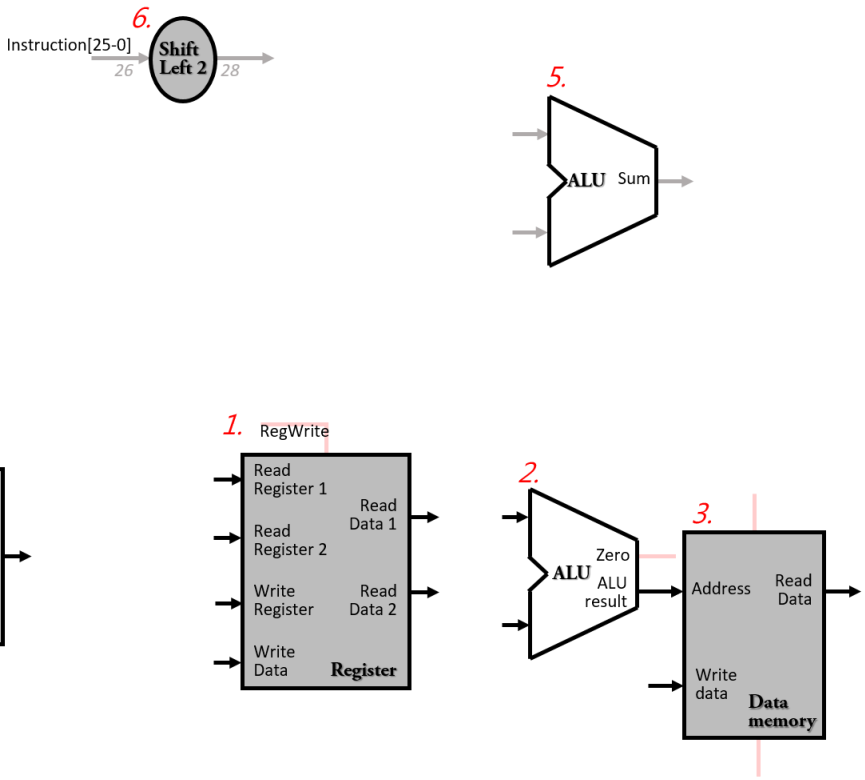
重點一：抽象化設計

抽象化設計是指在設計系統時，簡化較低層的細節，以降低設計的複雜度。課本將介紹三種版本的計算機

Specification	Instruction set								
	Single cycle machine		Multiple cycle machine			Pipeline(第五章)			
Machines	Datapath	Control unit	Datapath	Control unit		Datapath	Control unit	Hazards	Advance pipeline
				Hardwire	Micro programming				
Components	Instruction Memory, Data memory, Register file, ALU, Adders, PC, Sign extension unit, Logic gate, …								

基本零件：

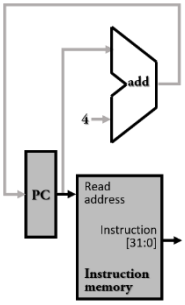
1. Register File：暫存器的讀寫
2. ALU：暫存器的運算
3. Data Memory：記憶體的讀寫
4. Instruction Memory：指令的讀取
5. Adder：下一個要執行的指令*(PC+4)*
6. Sign Extension Unit：數至位數擴充
7. PC：存放『下一個』要執行的指令位址



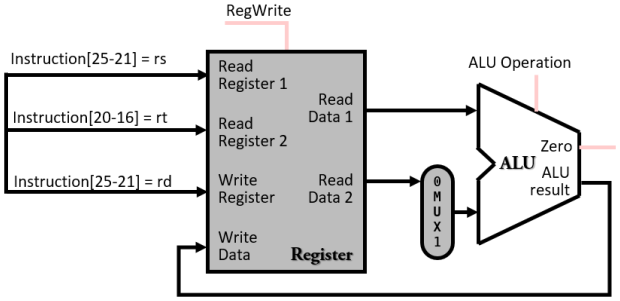
重點二：單一時脈計算機資料路徑的建構

Instruction fetch(指令擷取資料路徑)：RTL：PC <- PC+4

(RTL：暫存器轉移敘述 Register Transfer Language)

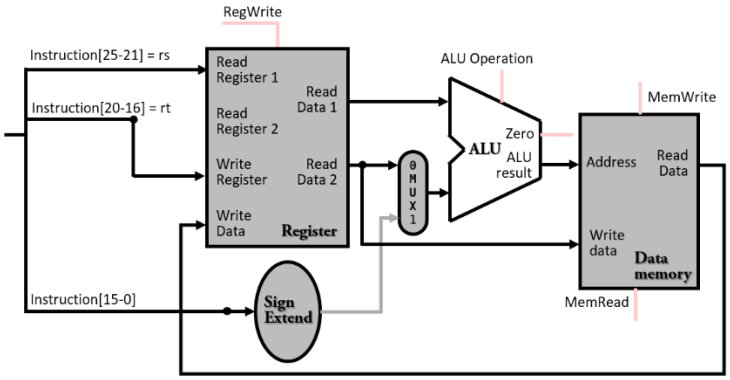


R-type：RLT：R[rd] <- R[rs] op R[rt]

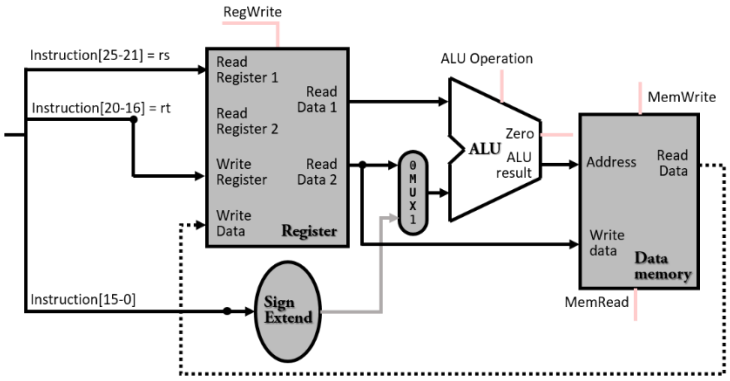


ALU Operation 有 4 個 bits 輸入：Ainvert、Binvert、Op1、Op2(ALUOp 輸入 ALU Control，輸出 4-bits 訊號，以組合成 and、or、add 和 slt)

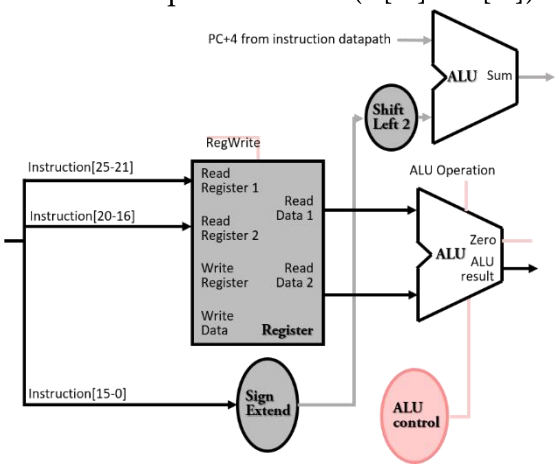
Load word : RTL : R[rt] <- Mem[R[rs] + SignExt(imm16)]



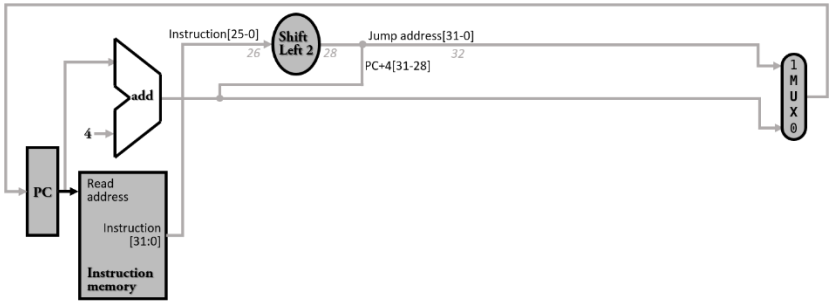
Store word : RTL : Mem[R[rs] + SignExt(imm16)] <- R[rt]



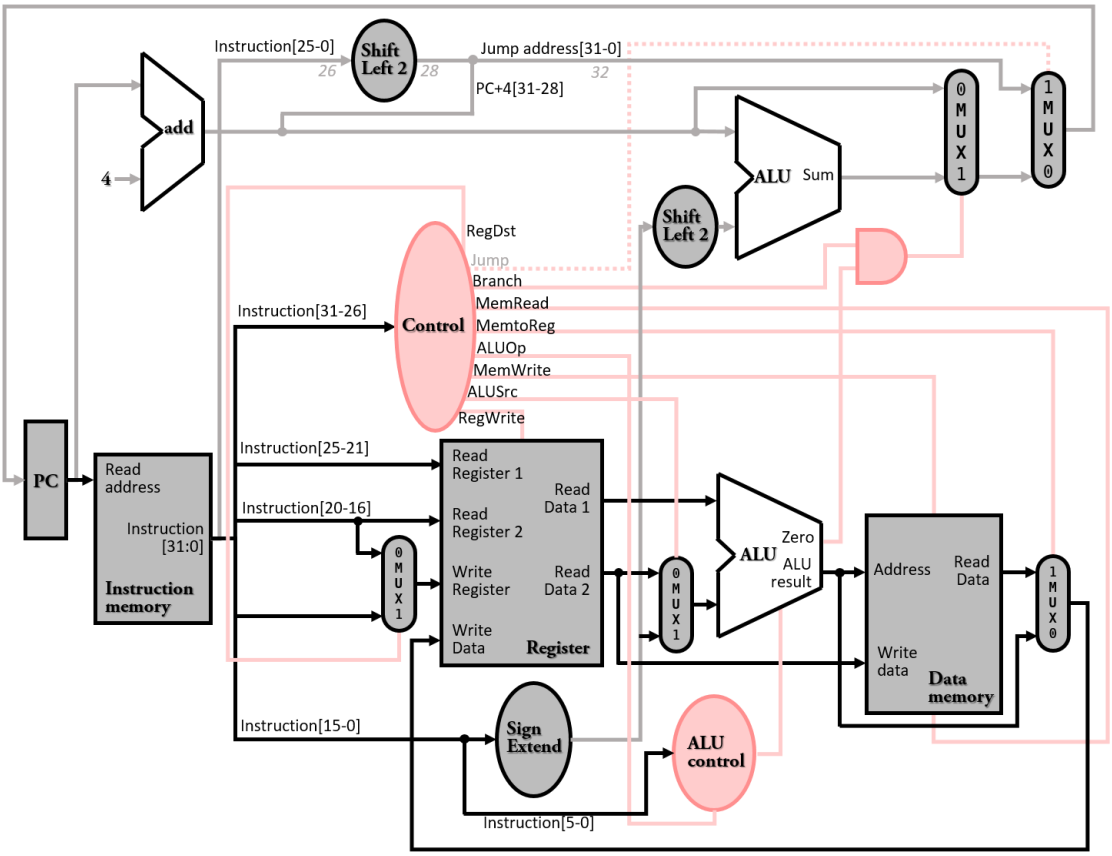
Branch on equal : RTL : if(R[rs] = R[rt]) PC <- (PC+4) + SignExt(imm16) << 2



Jump : RTL : PC <- {NextPC[31:28], Instruction[25:0], 2'b00}



全部合併且加上控制線路：



重點三：單一時脈計算機控制單元的建構

opcode[31-26]經過 Control 轉換為 9 條控制線(上圖紅色線)：

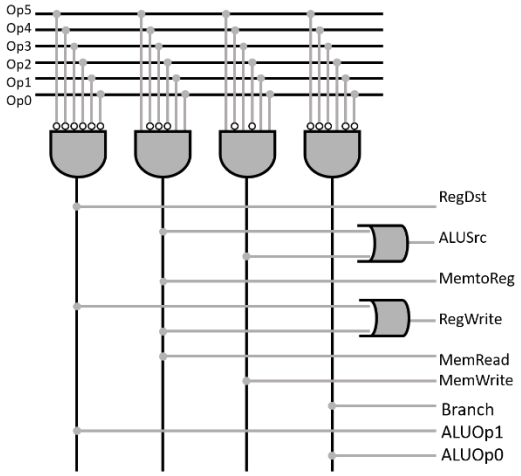
1. RegDst： ReadReg2 與 WriteReg
2. *Jump： Jump 與不 Jump(圖片中，1 在上方較好畫) (通常不算在內)*
3. Branch： 不 Branch 與 Branch
4. MemRead： MemRead 與否
5. MemtoReg：ReadData 與 ALUresult(圖片中，1 在上方較好畫)
6. ALUOp(2)：ALUcontrol (00：lw, sw、01：beq、10：R-type)
7. MemWrite：MemWrite 與否
8. ALUSrc： ReadData2 與 Signextend 結果
9. RegWrite： RegWrite 與否

不同指令所對應之控制值

	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	branch	ALUOp1	ALUOp0
R	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	x	1	x	0	0	1	0	0	0
beq	x	0	x	0	0	0	1	0	1
j	*x*	*x*	*x*	*0*	*0*	*0*	*0*	*x*	*x*

由此可得：6-bits Opcode 轉成 9 條控制線關係：

IO	Name	R-type	lw	sw	beq
Input	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Output	RegDst	1	0	x	x
	ALUsrc	0	1	1	0
	MemtoReg	0	1	x	x
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1



ALUOp 是由 6-bits function code 中解譯而來，之後控制 ALU Control、進而影響 ALU。需注意的部分是：ALUOp 並非 ALU 控制線中的 Op1、Op0：

指令	funct[5-0]	ALUOp	ALU 運算	ALU 控制線
lw	xxxxxx	00	add	0010
sw	xxxxxx	00	add	0010
beq	xxxxxx	01	sub	0110
add	100000	10	add	0010
sub	100010	10	sub	0110
and	100100	10	and	0000
or	100101	10	or	0001
slt	101010	10	slt	0111

因為 ALUOp 並無 11 之值，故可視為 00、x1 與 1x；又 function code 對 ALUOp 的影響僅 4-bits，故可得出：

指令	ALUOp		Function code						Operation
	ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
lw/sw	0	0	x	x	x	x	x	x	0010
beq	x	1	x	x	x	x	x	x	0110
add	1	x	x	x	0	0	0	0	0010
sub	1	x	x	x	0	0	1	0	0110
and	1	x	x	x	0	1	0	0	0000
or	1	x	x	x	0	1	0	1	0001
slt	1	x	x	x	1	0	1	0	0111

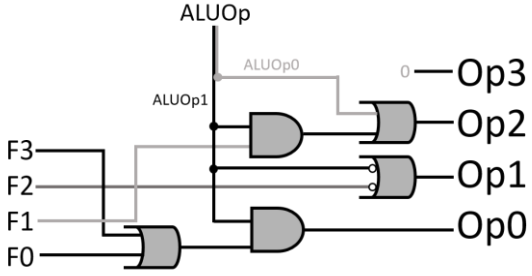
因此我們可以得到公式如下：

Op3 = 0

Op2 = ALUOp0 + (ALUOp1*F1)

Op1 = ALUOp1’ + F2’

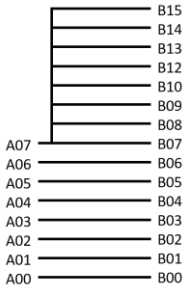
Op0 = ALUOp1 * (F0+F3)



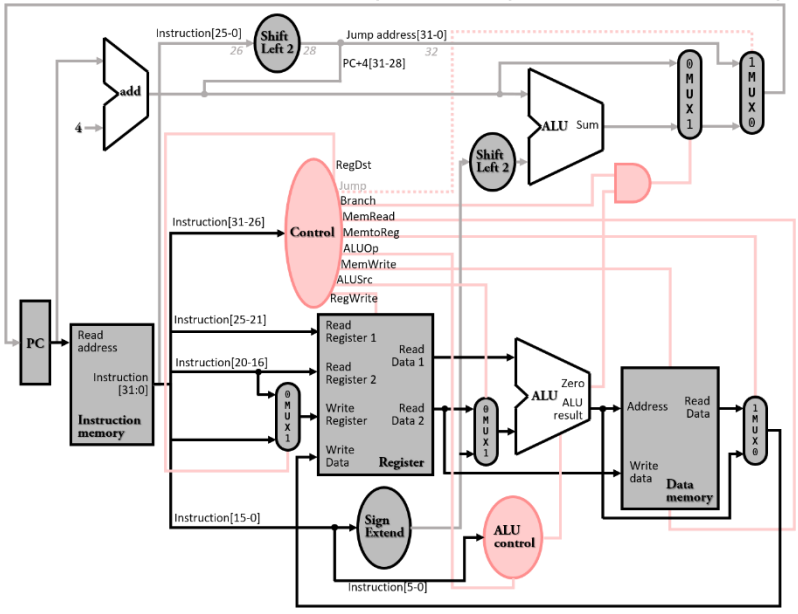
例(23)：Which of the following is NOT a register-transfer level component (RTL)?
1. register, 2. adder, 3. AND gate, 4. decoder

4

例(24)：Design a circuit which can convert 8-bit signed integer to 16-bit signed integer. The input is A7, A6,…,A0 (A7 is MSB). The output is B15, B14,…,B0 (B15 is the MSB).



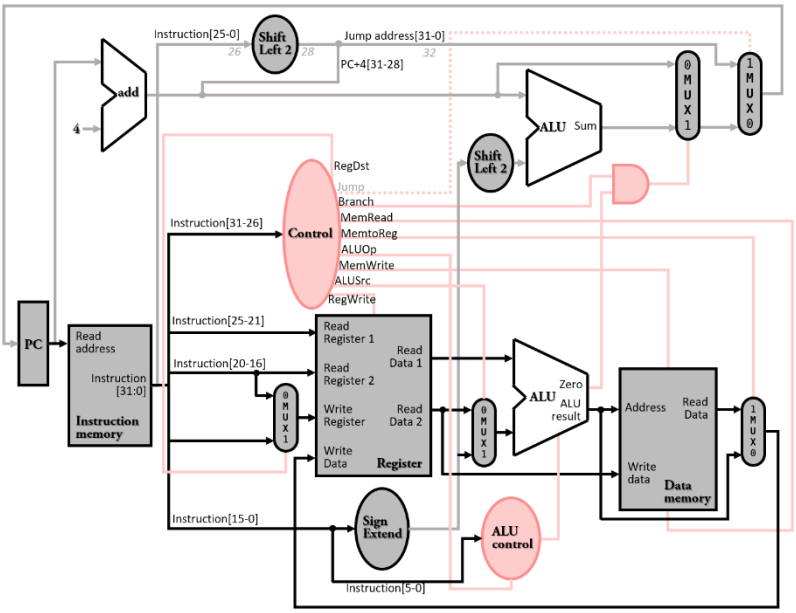
例(19)：Shown in the following Figure is a single-cycle implementation of the MIPS processor, answer the following questions.
1. What does “single-cycle” mean for this MIPS implementation?
2. What is the purpose of sending the value of PC register to an adder?
3. What is the purpose of using multiple levels of decoding tot generate the actual ALU control bits?
4. What is the purpose of performing “Shift left 2” on the output bits of “Sign extend”?
5. What is the purpose of sending the Zero signal of the ALU to AND gate?



1. *In a single-cycle machine, an instruction is executed in one clock cycle.*
2. *To increment the PC to obtain the address of the next sequential instruction (4 bytes latesr).*
3. *Using multiple levels of control can reduce the size of the main control unit and may also potentially increase the speed of the control unit.*
4. *To increases the effective range of the offset field of the branch instruction by a factor of 4.*
5. *Zero output of the ALU, which is used for equality comparison, will need to AND together the control signal, branch, to determine a branch instruction is taken or not.*

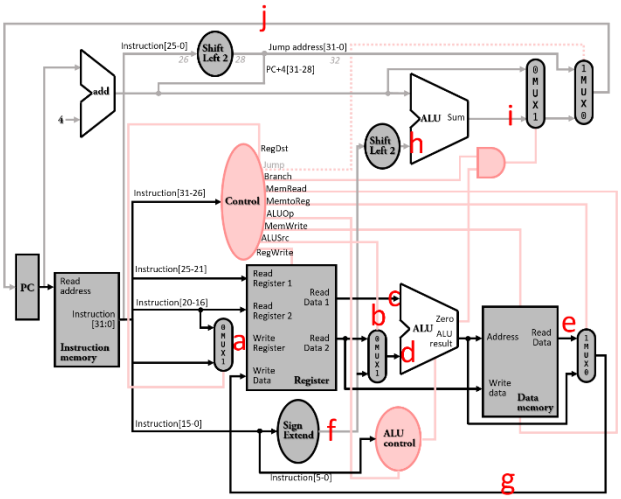
例(22)：(Refer to the CPU architecture of Figure 1 below) Which of the following statements is correct for a load word (LW) instruction?

1. MemtoReg should be set to 0 so that the correct ALU output can be sent to the register file.
2. MemotoReg should be set to 1 so that the Data Memory output can be sent to the register file.
3. We do not care about the setting of MemtoReg. It can be either 0 or 1.
4. MemWrite should be set to 1.



2

例(7)：Assume we want to execute the following addi instruction in the single-cycle datapath: addi $19. $29, 16. The single-cycle datapath diagram below shows the execution of this instruction.



1. Please set the control signals in the following table so that addi can be executed properly. Use 'x' for 'don't care' if necessary.

RegDst	ALUSrc	MemtoReg	PCSrc

2. Several of the datapath values are already shown in the diagram. You are to provide values for the ten remaining signals, which are labeled with a question mark (?) followed by a character (from a to j). Please write their decimal values in the following table. Assume register $29 and PC initially contain the value of 145 and 32, respectively. If a value cannot be determined, mark it as 'X'.

a	b	c	d	e	f	g	h	i	j

1.

RegDst	*ALUSrc*	*MemtoReg*	*PCSrc*
0	*1*	*0*	*0*

2.

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*
19	*x*	*145*	*16*	*x*	*16*	*161*	*64*	*100*	*36*

例(17)：Processor control

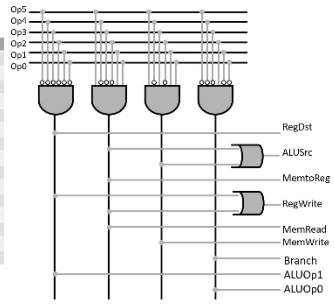
1. (刪掉十年以上)Please describe clearly and compare microprogramming and hardwired design for controlling a processor, in which one instruction is executed in several clock cycles.
2. Given that a processor supports four types of instructions: Arithmetic, Memory Load, Memory Store and Branch and that the following truth table specifies the inputs and outputs of the control mechanism, what control design strategy will you use and why? Please show the details of your design.

Contr	Input						Output						
Signal Name	In5	In4	In3	In2	In1	In0	Reg Dst	Reg Write	Mem Read	Mem Write	Branch	ALU Op1	ALU Op2
Arith.	0	0	0	0	0	0	1	1	0	0	0	1	0
Load	1	0	0	0	1	1	0	1	1	0	0	0	0
Store	1	0	1	0	1	1	x	0	0	1	0	0	0
Branch	0	0	0	1	0	0	x	0	0	0	1	0	1

1	*Microprogrammed control*	*Hardwired control*
解釋	*A method of specifying control that uses microcode rather than a finite state representation*	*An implementation of finite state machine control typically using programmable logic arrays (PLAs) or collections of PLAs and random logic*
優點	*1. 可簡化控制單元的設計* *2. 有彈性、修改設計容易* *3. 適合處理較複雜的指令集*	*執行速度快*
缺點	*1. 執行速度較慢* *2. 硬體成本較高(控制記憶體及解碼邏輯)*	*硬體設計上較為複雜、修改不易、適合處理較簡單的指令集*

2.

IO	Name	R-type	lw	sw	beq
Input	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Output	RegDst	1	0	x	x
	ALUsrc	0	1	1	0
	MemtoReg	0	1	x	x
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

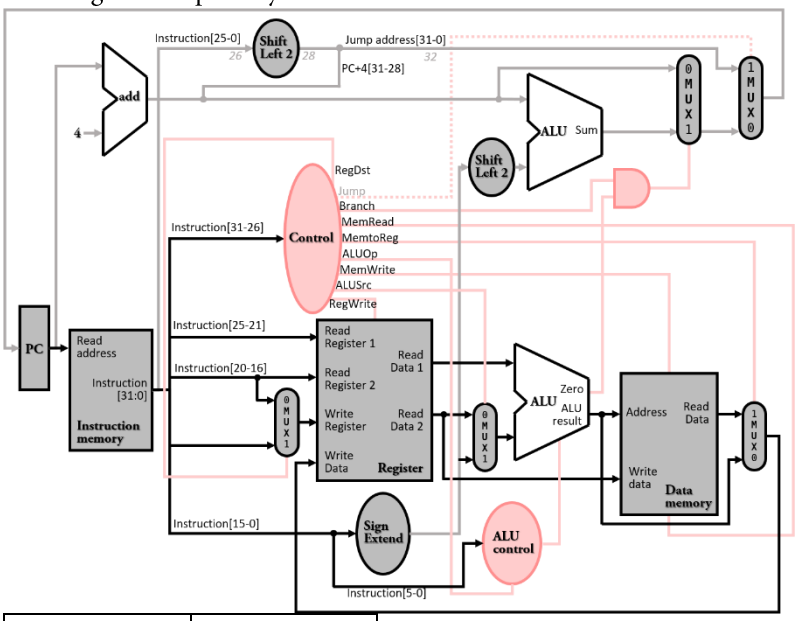


The structure, called a programmable logic array (PLA), uses an array of AND gates followed by an array of OR gates. The inputs to the AND gates are the function inputs and their inverses. The inputs to the OR gates are the outputs of the AND gates. The output of the OR gates is the function outputs.
PLA can simplify the design process and is one of the common ways to implement a control function.

例(14)：The table shown as follows is the main control of a single-cycle processor which cannot execute the j instruction. What are the values of RegDst, ALUSrc, MemtoReg, RegWrite, MemRead, Branch and MemWrite for executing the j instruction?

	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	branch	ALUOp1	ALUOp0
R	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	x	1	x	0	0	1	0	0	0
beq	x	0	x	0	0	0	1	0	1
j	*x*	*x*	*x*	*0*	*0*	*0*	*0*	*x*	*x*

練習：Describe the effect that a single stuck-at-0/1 fault (i.e., regardless of what it should be, the signal is always 0/1) would have for the signals shown below, in the single-cycle datapath in the following Figure. Which instructions, if any, will not work correctly? Explain why. Consider each of the following faults separately:



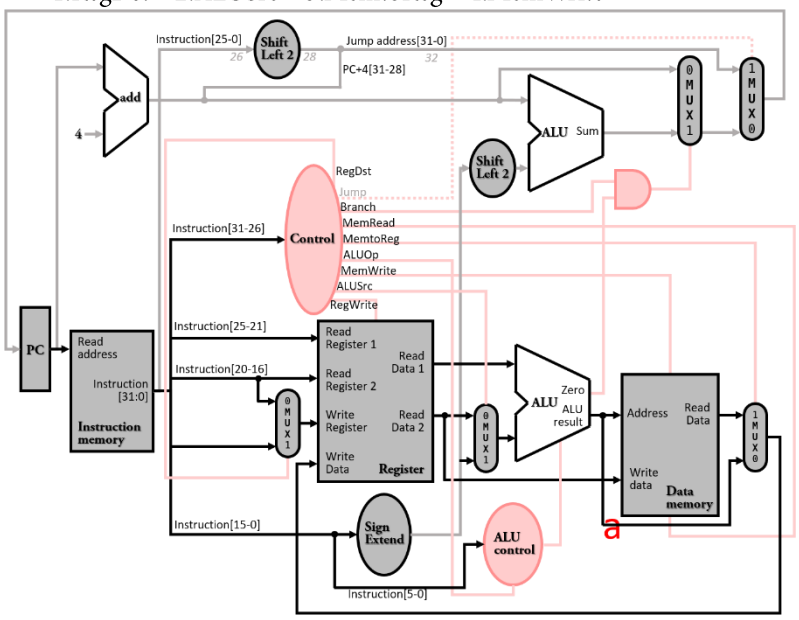
a.RegWrite=0	g.RegWrite=1
b.ALUOp1=0	h.ALUOp1=1
c.ALUOp0=0	i.ALUOp0=1
d.Branch=0	j.Branch=1
e.MemRead=0	k.MemRead=1
f.MemWrite=0	l.MemWrite=1

	Stack-at-0	*Stack-at-1*
RegWrite	*All R-type format instructions and lw will not work correctly.*	*Sw and beq will not work correctly.*
ALUOp1	*All R-type format instructions except addition will not work correctly.*	*Lw and sw will not work correctly.*
ALUOp0	*Beq will not work correctly.*	*Lw and sw will not work correctly.*
Branch	*Beq will not work correctly.*	*Instructions othe than branches (beq) will not work correctly.*
MemRead	*Lw will not work correctly.*	*All instructions will not work correctly.*
MemWrite	*Sw will not work correctly.*	*Only sw will work correctly.*

例(2-4)：Considering the code sequence, the single-cycle CPU below.

add $t2, $s1, $s2
sw $s1, 0($s2)
lw $t1, 0($s1)
	add $t2, $s2, $s3
	add $t3, $t2, $t2
	lw $t1, 0($t2)
		add $t1, $t1, $s1
		add $t1, $t1, $s2
		slt $t2, $s2, $t1

2. What instructions below may fail to run correctly if the datapath labeled A in the single-cycle CPU figure has been cut?
   1.add、2.sw、3.lw、4.slt
3. What instructions below may fail to run correctly if the control signal RegDst in the single-cycle CPU has been cut and stuck at 1?
   1.add、2.sw、3.lw、4.slt
4. With single-cycle CPU, what control signals below will be set to 1 when the code runs to the 5th cycle?
   1.RegDst、2.ALUSrc、3.MemtoReg、4.MemWrite



2. 1、4
3. 3
4. 1

重點四：單一時脈計算機的效能

練習：假設在單一時脈週期計算機中，各主要功能單元所需要的時間如下：
-記憶體單元：200 ps
-ALU 及加法器：100 ps
-暫存器檔案(讀取或寫入)：50 ps
假設多工器、控制單元、PC 存取、有號擴充單元和各單元之間的接線不花費任何時間。則下面哪一種實作方法比較快？快多少？

1. 每個指令執行時間固定花一個時脈週期的實作方式。
2. 每個指令執行時間花一個時脈週期，但是我們使用可變長度的時脈週期。也就是說每個指令僅僅需要它本身所需要花費的時間。

假設我們執行了下列混合比例的指令：25%載入指令、10%儲存指令、45%ALU 指令、15%分支指令，及 5%跳躍指令

我們只需要找出這兩種實作的時脈週期時間，因為這兩種實作的指令個數及 CPI 是相等的。不同類別指令的關鍵路徑(critical path)如下：

指令種類	*各類指令所使用之功能單元*				
R-type	*指令擷取*	*暫存器存取*	*ALU*	*暫存器存取*	
Lw	*指令擷取*	*暫存器存取*	*ALU*	*記憶體存取*	*暫存器存取*
Sw	*指令擷取*	*暫存器存取*	*ALU*	*記憶體存取*	
Branch	*指令擷取*	*暫存器存取*	*ALU*		
Jump	*指令擷取*				

利用這些關鍵路徑，我們可以計算執行不同類型指令所需要的時間長度：

Instruction	*IM*	*RegRead*	*ALUOp*	*DM*	*RegWrite*	*Total*	*Frequency*
R-type	*200*	*50*	*100*	*0*	*50*	*400ps*	*45%*
Lw	*200*	*50*	*100*	*200*	*50*	*600ps*	*25%*
Sw	*200*	*50*	*100*	*200*		*550ps*	*10%*
Branch	*200*	*50*	*100*			*350ps*	*15%*
Jump	*200*					*200ps*	*5%*

*對於使用單一時脈週期計算機而言，時脈週期由最長的指令決定，也就是 600ps。對於可變時脈的計算機而言，我們可以找到可變時脈計算機的平均時脈週期長度：Instruction time=600*25% + 550*10% + 400*45% + 350*15% + 200*5% = 447.5*

因為可變時脈的實作方式有較短的平均時脈週期，這種實作明顯的比較快，接下來比較其效能：

$$CPUperformance_{variable\ clock} / CPUperformance_{single\ clock} = 600/447.5 = 1.34$$

例(30)：Given the datapath for the single-cycle implementation of a computer and the definition of its instructions: add, lw, sw, beq

Memory unit: 2ns、ALU and adders: 2ns、Register file: 1ns、其他無 delay

Instruction Class	*Instruction Memory*	*Register (Read)*	*ALU*	*Data Memory*	*Register (Write)*	*Total Time*
add	*2*	*1*	*2*		*1*	*6 ns*
lw	*2*	*1*	*2*	*2*	*1*	*8 ns*
sw	*2*	*1*	*2*	*2*		*7 ns*
beq	*2*	*1*	*2*			*5 ns*

The execution time for each instruction is 8 ns since the clock cycle for the single cycle machine is determined by longest instruction, which is 8 ns.

練習：In this exercise we examine how latencies of individual components of the datapath affect the clock cycle time of the entire datapath, and how these components are utilized by instructions. For problems in this exercise, assume the following latencies for logic blocks in the datapath.

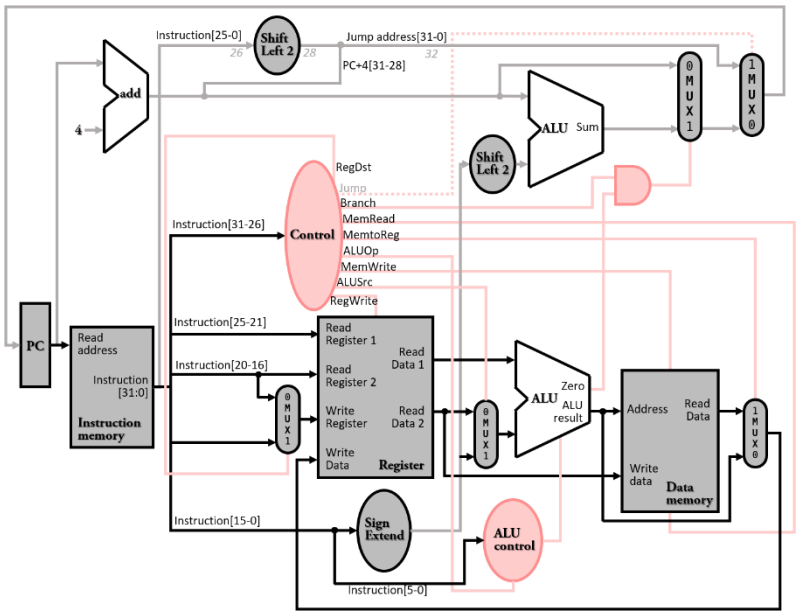
I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-extend	Sift-left-2
500ps	150ps	100ps	180ps	220ps	1000ps	90ps	20ps

1. What is the clock cycle time if the only type of instructions we need to support are ALU instructions (add, and, etc.)?
2. What is the clock cycle time if we only had to support lw instructions?
3. What is the cloc cycle time if we must support add, beq, lw, and sw instructions?

1. *The longest-latency path for ALU operations: I-Mem -> Regs -> Mux -> ALU -> Mux -> Regs. So we have: 500+220+100+180+100+220=1320 ps*
2. *The longest-latency path for lw operations: I-Mem -> Regs -> ALU -> D-Mem -> Mux -> Regs. So we have: 500+220+100+180+1000+100+220=2220 ps*

3. Because the lw instruction has the longest critical path, the clock cycle time is 2220 ps.

例(11)：Given the datapath for a single-cycle implementation of a computer and the definition and formats of its instructions as follows:



Assume that control signals ALUOp = 00 for performing addition of the ALU unit, ALUOp = 01 for sub, and ALUOp = 10 while depending on the funct field of the instruction.

1. Assume that logic blocks needed to implement the datapath have the following latencies: (Delays for other components are ignored.)

I-Mem	Add	Mux	ALU	Regs	D-Mem	Signextend	Shift-left-2
400	100	40	120	200	350	200	10

   Compute the required delay time for instruction and determine the minimum cycle time of the computer.
2. Specify the values of the control signals for instructions add, lw, and, beq. Express your answere as the following table: (Ddnote a don't-care control signal as "x".)

1.

	I-Mem	*Regs*	*Mux*	*ALU*	*D-Mem*	*Mux*	*Regs*	*Delays*
add	*400*	*200*	*40*	*120*		*40*	*200*	*1000*
lw	*400*	*200*		*120*	*350*	*40*	*200*	*1310*
sw	*400*	*200*		*120*	*350*			*1070*
beq	*400*	*200*	*40*	*120*		*40*		*800*

2.

	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	branch	ALUOp1	ALUOp0
R	*1*	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*0*
lw	*0*	*1*	*1*	*1*	*1*	*0*	*0*	*0*	*0*
beq	*x*	*0*	*x*	*0*	*0*	*0*	*1*	*0*	*1*

例(6)：In the basic single-cycle implementation of the following Figure, there are seven kinds of major blocks. Latencies of blocks along the critical (longest-latency) path for an instruction determine the minimum latency of that instruction. Assume the following resource latencies:

1. What is the latency of the critical path for a MIPS AND instruction?
2. What is the latency of the critical path for a MIPS lw instruction?
3. What is the latency of the critical path for a MIPS beq instruction?

1. *400+200+30+120+30+200=980 ps*
2. *400+200+120+350+30+200=1300 ps*
3. *400+200+30+120=750 ps*

例(26)：(Circle all correct answers) In a single-cycle datapath design of the MIPS architecture, which of the following descriptions are correct?

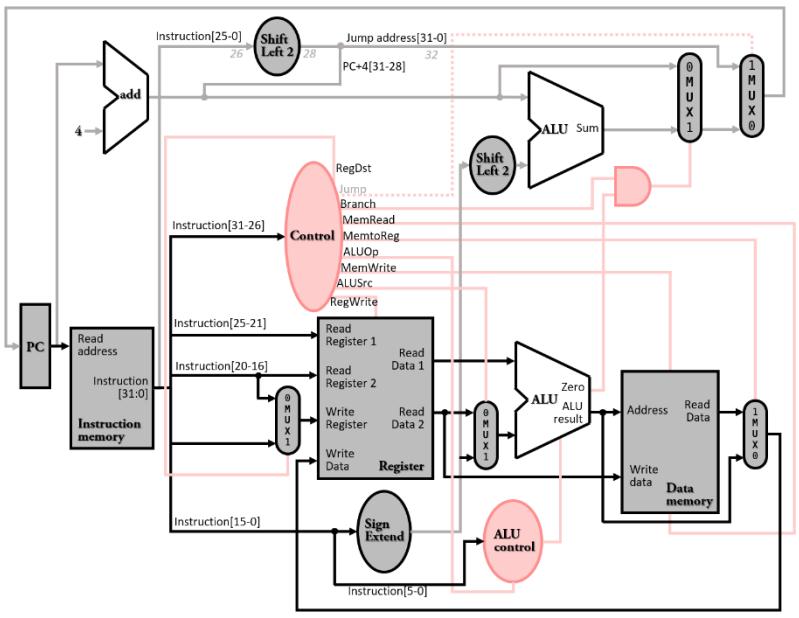
1. The data flow of R-type instructions does not go through the data mem.'
2. The data flow of SW goes through all components in a clock cycle.
3. The data flow of LW goes through all components at most once in a clock cycle.
4. The data flow of J-type instructions goes through all components.

1、2

例(25)：Design a single-cycle implementation of processor for the following MIPS-like instructions: lw Rt, offset(Rs), sw Rt, offset(Rs), add Rd, Rs, Rt, and beq Rs, Rt, label. Answer the following questions:

1. Show the datapath and control of this processor.
2. How to control the write for the data memory? Should it be edge-triggered or not? Why?

1.



2.

The steps to control the write for the data memory are as follows:

1. *Apply the address calculated by the ALU to the address inputs*
2. *Apply the data from the register file to the write data inputs*
3. *Assert the MemWrite control line.*

The data memory should be edge-triggered since the write data inputs and the address input of the data memory may change their values twice during a single clock cycle.

練習：Assume that the logic blocks used to implement the datapath have the following latencies: (this problem is based on the single-cycle machine)

	I-Mem	Add	Mux	ALU	Regs	D-Mem	Signextend	Shift-left-2	ALUCtrl
a.	400	100	30	120	200	350	20	0	50
b.	500	150	100	180	220	1000	90	20	55

1. To avoid lengthening the critical path of the datapath of single-cycle machine, how much time can the control unit take to generate the MemWrite signal.
2. Which control signal has the most slack and how much time does the control unit have to generate it if it wants to avoid being on the critical path?
3. Which control signal is the most critical to generate quickly and how much time does the control unit have to generate it if it wants to avoid being on the critical path?

Assuming zero latency for the Control unit, the critical path is the path to get the data for load instruction.

1. *The Control unit can begin generating MemWrite only after I-Mem is read. It must finish generating this signal before the end of the clock cycle. So the Control unit must generate the MemWrite in one clock cycle, minus the I-Mem and D-Mem access time:*

	Critical path	*Maximum time to generat MemWrite*
a.	*400+200+120+350+30+200=1300*	*1300-400-350=550*
b.	*500+220+180+1000+100+220=2220*	*2220-500-1000=720*

2. *All control signals start to be generated after I-Mem read is complete and jump instruction is finished as soon as jump target going through MUX, which is controlled by Jump control signal. Since MUX has less latency, Jump control signal has the most slack.*

a.	*b.*
1300-400-30=870	*2220-500-100=1620*

3. *ALUOp and ALUSrc would be the most critical to generate quickly. ALUOp is more critical than ALUSrc if ALU control has more latency than a MUX. If ALUOp is the most critical, it must be generated one ALU Ctrl latency before the critical-path signals. If the ALUSrc signal is the most critical, it must be generated while the critical path goes through MUX.*

	The most critical control signal is	*Time to generate it without affecting the clock cycle time*
a.	*ALUOp (50>30)*	*200+30-50=180*
b.	*ALUSrc (100>55)*	*220*

重點五：多重時脈計算機

指令	Step name					CPI
R-type	IF	ID/RF	Exe		WB	4
lw	IF	ID/RF	Addr. calculation	MA	WB	5
sw	IF	ID/RF	Addr. calculation	MA		4
beq	IF	ID/RF	Compare reg.			3
j	IF	ID	Compose target addr.			3
Functional unit	Instruct. Mem.	Reg. file	ALU	Data Mem.	Reg. file	

1. Instruction Fetch(ID)
2. Instruction Decode / Register Fetch(ID/RF)
3. Execution, Memory Address Computation, or Branch Completion(ALU)
4. Memory Access or R-type instruction completion(MA)
5. Memory Read Completion(WB)

例(21)：Consider an instruction set is implemented with the multi-cycle architecture. The instructions are classified into 4 classes, i.e. calculation, branch, memory read, and memory write/ The stages of instruction execution and the execution times of each stage are shown in the following table:

	Instruction fetch	Decode and fetch operands	ALU	Memory read/write	Write back
Calculation	O	O	O		O
Branch	O	O	O		
Memory read	O	O	O	O	O
Memory write	O	O	O	O	
ExTime	60 ns	20 ns	40 ns	60 ns	25 ns

Consider a program executing on this machine. The distribution of instructions is shown below.

Calculation	Branch	Memory read	Memory write
47%	20%	21%	12%

Calculate the average execution time for an instruction.

	Calculation	*Branch*	*Memory read*	*Memory write*
Instruction Mix	*47%*	*20%*	*21%*	*12%*
CPI	*4*	*3*	*5*	*4*
Cycle time	*60 ns*			
Average CPI	*0.47*4 + 0.2*3 + 0.21*5 + 0.12*4 = 4.01*			
Instruction time	*4.01*60 ns = 240.6*			

例(9)：Multicycle Datapath:

```
lw    $t2, 0($t3)
lw    $t3, 4($t3)
beq   $t2, $t3, Label     :assume not equal
add   $t5, $t2, $t3
sw    $t5, 8($t3)
```

What is going on during the 6th and 12th cycle of execution?

1. *Fetch instruction "lw $t3, 4($t3)" from memory*
2. *Decode instruction "beq $t2, $t3, Label" and read registers, $t2 and $t3, from register file.*

例(10)：A benchmark is used to design a microprocessor called LHOR with five instruction classes and their dynamic statistics is as follows:

Class	Statistics
ALU instructions	50%
Loads	20%
Stores	14%
Branches	12%
Jumps	4%

Two implementations are compared, i.e., the single-cycle implementation and multi-cycle implementation. The critical path, which is 60 ns, exists in the Load-type instruction class for the single-cycle implementation. The multi-cycle design is done by partitioning the Load-type instruction to 6 stages, reducing the cycle period. Assume that the pipeline register introduces additional 2 ns delay to each stage. Adopting the multi-cycle implementation, the number of clock cycles for each instruction class is listed as follows:

Class	Loads	Stores	ALU	Branches	Jumps
Cycles	6	5	4	4	3

1. What is the ratio between numbers of execution cycles for the multi-cycle implementation and the single-cycle one based on this benchmark?
2. What is the performance speedup of the multi-cycle design?

1. *CPI for multi-cycle machine = 6*0.2 + 5*0.14 + 4*0.5 + 4*0.12 + 3*0.04 = 4.5. The ratio = 4.5/1 = 4.5*
2. *Speedup = 1*60ns / 4.5*12ns = 1.11*

例(13)：Consider a computer architecture with an instruction set which is composed of N different instruction classes. The execution time of class-I instruction is 4+I ns, where 1<= i<= N.

1. If the computer is implemented with a single-cycle design, what is the minimum clock cycle time? Why?
2. Consider a multicycle control implantation with a clock cycle of 1 ns. Ignore the overhead associated with multicycle control. What is the performance advantage (speedup) of multicycle control relative to single-cycle control, assuming that the various instruction classes are used with the same frequency?
3. With the performance benefit function derived in part 2, would a speedup factor of 5 be possible? What would be the requirement to achieve that speedup?
4. Repeat part 2 for the case when the relative frequency of class-I instruction is proportion to i.
5. Repeat part 2 for the case when the clock cycle time is 2 ns and the class-I instruction requires (4+i)/2 clock cycles. Assume N = 5 in this case.

1. *The minimum clock cycle time is s(4+N) ns since in a single-cycle design, clock cycle time is determined by the longest execution time instruction.*
2. *Instruction time for single cycle machine = (4+N) ns. Instruction time for multicycle machine = 1*Sum(4+i) / N = (4.5+0.5N) ns ; speedup = 4+N / 4.5+0.5N*
3. *4+N / 4.5+0.5N = 5 => N=-12.33. A speedup of 5 is impoosible unless pipeline technique is used.*
4. *Instruction time for mulicycle machine = 1 * Sum(4+i) *I / N(N+1)/2 = 4 + (2N+1)/3 ns ; Speedup=4+N / 4 + (2N+1)/3 = 12+3N / 13+2N*
5. *Instruction time for single cycle machine = 4+5=9ns Instruction time for multicycle machine = 2 * Sum(4+i)/2 / 5 = 7ns ; Speedup = 9/7=1.2857*