

학부		학번		반		이름	
----	--	----	--	---	--	----	--

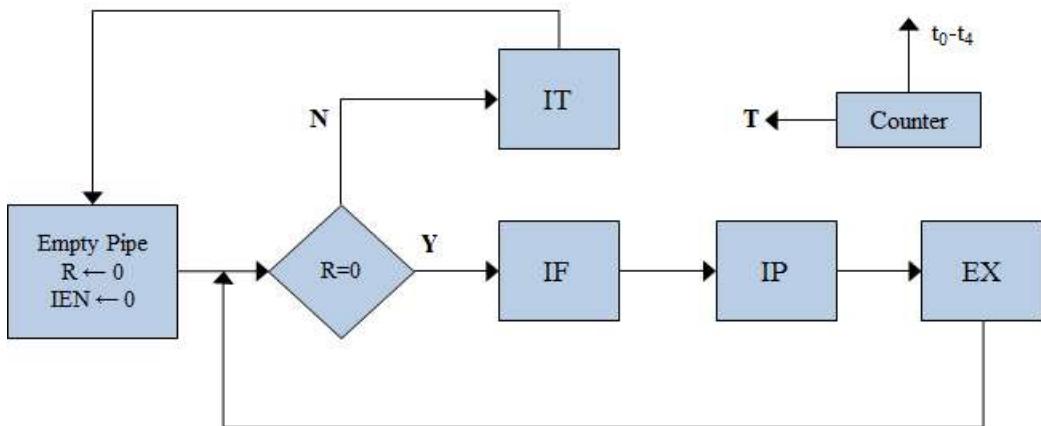
1. Mano의 Basic Computer에서는 인터럽트 서비스가 실행되는 동안 또 다른 인터럽트를 받을 수 없다. 이를 해결하기 위하여 인터럽트 서비스 이후에 돌아갈 리턴 주소와 AC 및 E 레지스터의 내용을 스택에 저장하는 구조로 변경할 때, 다음 물음에 답하라. 스택은 데이터의 마지막에 저장된 데이터가 먼저 나오는 방식의 데이터 구조이다. 이 때 저장될 위치를 알려주는 값을 스택포인터가 가지고 있다.

- 가. 스택은 메모리의 4001번지부터 시작되고,
- 나. 스택 포인터는 메모리 4000번지에 저장된다.
- 다. 인터럽트 서비스 루틴이 호출되는 경우, 일단 메모리 0번지에 리턴 주소를 저장하고, 스택에는 Flag E와 AC 값 및 리턴 주소가 저장된다. 돌아올 때는 스택의 Flag E와 AC 값을 복원하고, 리턴주소는 메모리 0번지에 저장하여서 돌아오게 된다. 다음의 인터럽트 서비스 루틴의 빈 곳을 채워라.(각 2점)

LINE	프로그램			LINE	프로그램		
1		ORG	0	24		BSA	⑤ DSP
2	ZRO,	HEX	0	25		LDA	SP I
3		BSA	PSH	26		⑥ CIL	
4		SKI		27		BSA	DSP
5		BUN	OUT	28		LDA	SP I
6		BUN	INP	29		ION	
7	PSH,	HEX	0	30		BUN	POP I
8		IOF		31	DSP,	HEX	0
9		STA	SP I	32		LDA	SP
10		ISZ	SP	33		⑦ ADD	ONE
11		① CIR		34		STA	SP
12		STA	SP I	35		BUN	DSP I
13		ISZ	SP	36	⑧ ONE	DEC	-1
14		LDA	② ZRO	37	INP,	INP	
15		STA	SP I	38		OUT	
16		ISZ	SP	39		BSA	⑨ POP
17		③ ION		40		BUN	ZRO I
18		BUN	PSH I	41	OUT,	OUT	
19	POP,	HEX	0	42		BSA	POP
20		IOF		43		BUN	⑩ ZRO I
21		BSA	DSP	44		ORG	4000
22		LDA	SP I	45	SP,	DEC	4001
23		④ STA	ZRO				

\* ① ↔ ⑥, (즉, ① CIR ⑥ CIL or ① CIL ⑥ CIR)

2. 다음 그림은 Mano가 설계한 컴퓨터를 위한 instruction pipeline의 제어부분으로 instruction fetch(IF)와 indirect processing(IP), execution(EX)의 세 단계로 구분되고 IF 단계는 interrupt processing(IT)와 정상처리부분(IF)로 구분된다. 전체 pipeline을 동기화하기 위한 time signal T와 각 단계 내부의 time signal  $t(t_0 \sim t_4)$ 가 있다고 가정한다. 표는 ISZ instruction을 예로 각 단계를 위한 수정된 RTL이다.

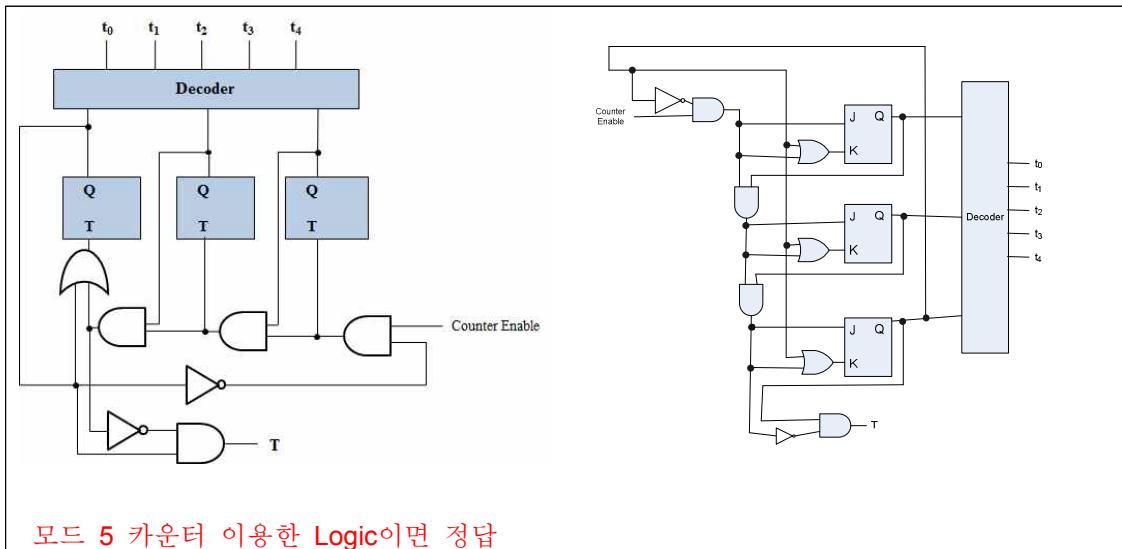


	IF	IT	IP	EX(ISZ)
$t_0$	No-operation	No-operation	D7'I: <input type="text"/> (D7'I)': AR $\leftarrow$ AR1	No-operation
$t_1$	$AR1 \leftarrow PC$	$AR1 \leftarrow 0,$ $TR \leftarrow PC$	No-operation	$DR \leftarrow M[AR]$
$t_2$	$IR \leftarrow M[AR1],$ $PC \leftarrow PC + 1$	$M[AR1] \leftarrow TR$ $PC \leftarrow 0$	No-operation	$DR \leftarrow DR + 1$
$t_3$	$D0\cdots D7 \leftarrow dec(IR(12-14)),$ $AR1 \leftarrow IR(0-11),$ $I \leftarrow IR(15)$	$PC \leftarrow PC + 1$	No-operation	$M[AR] \leftarrow DR,$ IF( $DR=0$ ) then $(PC \leftarrow PC + 1)$
$t_4$	No-operation	No-operation	No-operation	

(1) AR1 register를 따로 두어야 하는 이유를 설명하라.

**EX** 수행 동안 IF가 일어나기 때문에 IF과정에서 AR1을 이용하고 **EX**는 AR을 사용한다.

(2) 이 pipeline의 동작을 동기화하기 위한 counter register는 각 단계 내부의 time signal  $t_0 \dots t_4$ 를 생성하고,  $t_4$ 가 끝날 때마다 생성되는 time signal은 각 단계의 동기화에 이용된다. 이를 위한 counter logic을 설계하라.



(3) IP 단계의  $t_0$ 에 있는 빈칸을 채워라.

$AR \leftarrow M[AR1]$

(4) IF, IT, EX 단계의  $t_0$ 에 no-operation이 들어가는 이유를 설명하라.

Data dependency를 피하기 위해, 데이터 충돌을 막기 위해,  
Pipeline을 위한 Delayed load의 방법으로 No-operation 삽입 등 적절한 이유 시 정답

(5) EX 단계의  $t_4$ 에 빈칸을 메워라.

$(IEN)(FGI+FGO): R \leftarrow 1$

3. 다음 프로그램은 원소가 4개인 배열 A와 배열 B를 더하여서 배열 C에 저장하는 서브루틴이다. 배열의 시작주소와 원소의 개수는 파라미터로 서브루틴에 전달된다.

주소				주소		
4		BSA	MAT	6B		ISZ PT2
5		HEX	100	6C		ISZ PT3
6		HEX	200	6D		ISZ CTR
7		HEX	300	6E		BUN LOP
8		DEC	-4	6F		BUN MAT I
9		HLT		70	PT1	DEC 0
				71	PT2	DEC 0
		ORG	50	72	PT3	DEC 0
50	MAT,	HEX	0	73	CTR	DEC 0
51		LDA	MAT I			ORG 100
52		STA	PT1	100		DEC 6
53		ISZ	MAT	101		DEC 4
54		LDA	MAT I	102		DEC 7
55		STA	PT2	103		DEC 9
56		ISZ	MAT			ORG 200
57		LDA	MAT I	200		DEC 4
58		STA	PT3	201		DEC 6
59		ISZ	MAT	202		DEC 3
5A		LDA	MAT I	203		DEC 1
5B		STA	CTR			ORG 300
5C		ISZ	MAT	300		DEC 0
5D	LOP,	LDA	PT1 I	301		DEC 0
5E		ADD	PT2 I	302		DEC 0
5F		STA	PT3 I	303		DEC 0
6A		ISZ	PT1			

해당번지의 instruction0이 실행된 다음 레지스터와 메모리의 결과를 작성하라.[각 2 점]

명령어	4	54	6F
AC		200 <sub>(16)</sub>	
PC	51 <sub>(16)</sub>	55 <sub>(16)</sub>	9 <sub>(16)</sub>
50번지	5 <sub>(16)</sub>	6 <sub>(16)</sub>	9 <sub>(16)</sub>
70번지		100 <sub>(16)</sub>	104 <sub>(16)</sub>
302번지			10 <sub>(10)</sub>

동일한 16진수 or 10진수 값도 정답

4. 다음 설명들에 대하여 O/X로 답하라(1~10) [단 맞으면 2점, 틀리면 1점 감점]

- (1) Addressing을 하지 않는다면, assembler는 single pass로 가능하다.
- (2) Zero-address instruction의 경우, stack이 없으면 구현이 불가능하다.
- (3) Floating point 숫자의 덧셈연산을 pipeline으로 할 때, normalization이 필요한 이유는 mantissa field의 범위가  $0.5 \leq \text{mantissa} < 1$ 을 유지하여야 하기 때문이다.
- (4) Assembly 과정을 마치면 Pseudo-instruction은 기계어 프로그램에서 없어진다.
- (5) 1 MIPS의 CPU와 10 CPS의 I/O장치 간에 interrupt handling을 하지 않는 경우 CPU waste time은 거의 1 second에 접근한다.
- (6) Indirect addressing을 통하여 instruction 크기를 확장하지 않고, 더 큰 메모리 를 access할 수 있다.
- (7) Floating point 연산을 arithmetic pipeline으로 병렬 처리하는 경우, 처리해야 할 연산의 수가 pipeline segment 보다 적으면 병렬처리 효과가 없다.
- (8) Horizontal instruction type의 경우 decoding 없이 명령어를 수행할 수 있다.
- (9) 인터럽트를 처리하기 위해서는 반드시 인터럽트 발생하기 전에 수행하던 프로그램에서 참조한 모든 레지스터들을 저장하고, 돌아오기 전에 복원하여야 한다.
- (10) DMA 전송에서 입출력이 진행되는 동안에 CPU는 메모리 연결된 버스의 제어 를 할 수 없다.

1	O	2	O	3	O	4	O	5	O
6	O	7	O	8	O	9	O	10	O

5. 다음 설명에 해당하는 용어를 아래에 제시된 용어들 중에서 선택하라. [각 2점]

- (1) 메모리를 참조하는 명령어(memory reference instruction)에서 실제 데이터가 있는 메모리의 주소 [ effective address ]
- (2) Subroutine이 호출(call)될 때 돌아올 주소를 저장하기 위한 특정한 하나의 장소 [ save area ]
- (3) Interrupt가 발생하였을 때, 해당 인터럽트를 위하여 수행해야 할 프로그램 모듈 [ interrupt service program ]
- (4) 프로그램이 수행하는 동안 발생할 수 있는 오류(Error)를 처리하기 위하여 추가되는 프로그램 모듈 [ exceptional handling ]
- (5) 어떤 컴퓨터의 명령어 집합만으로 모든 문제를 프로그램으로 만들 수 있는 경우, 이런 컴퓨터 명령어 집합의 성질 [ instruction set complete ]
- (6) 명령어 종류를 줄이고, addressing mode를 최소화하여 instruction pipeline의 수행이 적합하게 설계되는 컴퓨터의 설계방식 혹은 그런 종류의 컴퓨터 [ RISC ]
- (7) 동시에 여러 개의 Interrupt들이 발생한 경우, 인터럽트에 대한 우선순위를 고정적으로 정하는 방식 [ priority interrupt ]

(8) 두 개의 명령어가 동시에 수행 될 수 없는 여러 상황 중에 하나의 명령어가 생성한 데이터를 다른 명령어가 사용하려는 경우, 두 명령어의 관계를 정의하는 성질

[ data dependence ]

(9) Instruction pipeline을 수행할 때, PC(program counter)를 예측할 수 없어서 pipeline의 작업을 정상으로 진행 할 수 없는 경우

[ branch conflict ]

(10) 컴퓨터에 연결된 입출력장치가 여러 개 인 경우, 각 정치들의 보내는 인터럽트들이 발생할 때 처리할 프로그램이 있는 장소들을 모아둔 메모리의 리스트

[ interrupt vector ]

anti-dependence, arithmetic pipelining, CISC, daisy-chaining, data dependence, double precision, effective address, exceptional handling, indirect address, instruction cycle, interrupt service program, instruction set complete, interrupt vector, memory interleaving, multi-core architecture, output dependence, priority interrupt, resource conflict, return address, RISC, save area, stored program concept

6. 다음 3개 중 Mano의 basic computer에서 single clock pulse 내에 수행될 수 없는 것을 모두 선택하고, 선택한 작업을 수행하기 위한 micro-operation sequence를 서술하시오. (4점)

- (1)  $IR \leftarrow M[PC]$
- (2)  $AC \leftarrow DR$ ,  $DR \leftarrow AC$
- (3)  $IR \leftarrow IR + 1$

(1)  
 $AR \leftarrow PC$   
 $IR \leftarrow M[AR]$

(3)  
 $DR \leftarrow IR$   
 $DR \leftarrow DR + 1$   
 $IR \leftarrow DR$   
(DR 대신 TR을 사용해도 무관)

(2)을 선택 시 0.5점 감점

7. DSZ라는 memory-reference instruction을 설계하여 ISZ를 대체하려 한다. DSZ는 ISZ와 반대로 주어진 메모리 주소에 있는 값을 1 감소시킨 다음 그 메모리 주소로 다시 저장하고 감소시킨 값이 0이면 다음 instruction을 skip한다. (주의: 명령어 실행 후 본래 AC에 저장되어 있던 값이 변경되어서는 안 된다) (6점)

[※ Ti가 더 필요한 경우 아래쪽에 추가하여 작성할 수 있다 (최대 T15까지) ]

D6T4: TR  $\leftarrow$  AC  
D6T5: DR  $\leftarrow$  M[AR], AC  $\leftarrow$  0  
D6T6: AC  $\leftarrow$  complement AC  
D6T7: AC  $\leftarrow$  AC + DR  
D6T8: M[AR]  $\leftarrow$  AC, if (AC = 0) then (PC  $\leftarrow$  PC + 1)  
D6T9: DR  $\leftarrow$  TR  
D6T10: AC  $\leftarrow$  DR, SC  $\leftarrow$  0

또는

D6T4: DR  $\leftarrow$  M[AR]  
D6T5: AC  $\leftarrow$  DR, DR  $\leftarrow$  AC  
D6T6: AC  $\leftarrow$  complement AC  
D6T7: AC  $\leftarrow$  AC + 1  
D6T8: AC  $\leftarrow$  complement AC  
D6T9: M[AR]  $\leftarrow$  AC, if (AC = 0) then (PC  $\leftarrow$  PC + 1)  
D6T10: AC  $\leftarrow$  DR, SC  $\leftarrow$  0