2014년 1학기 컴퓨터구조 기말고사

학과		학번		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 데이터구조이다. 이 때 저장될 위치를 알려주는 값을 스택포인터가 가지고 있다.

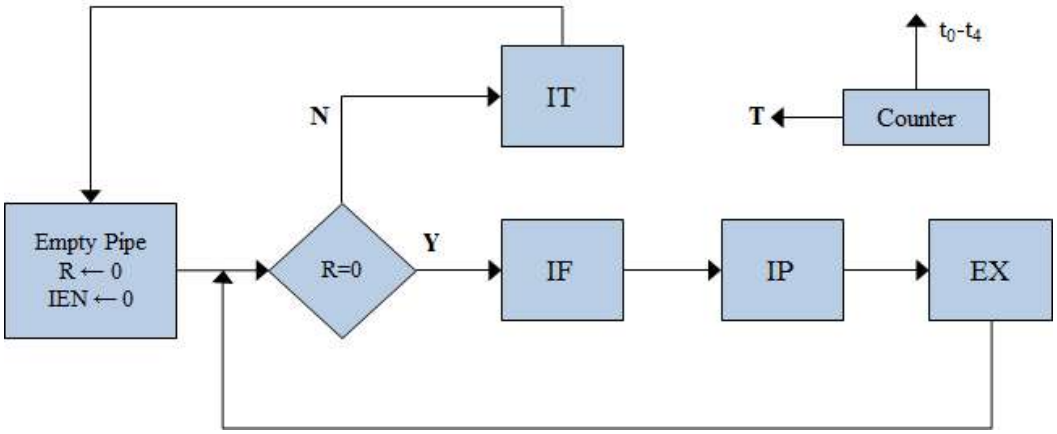
가. 스택은 메모리의 4001번지부터 시작되고,

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. **[각1점]**

LINE	PROGRAM			LINE	PROGRAM		
1		ORG	0	26		BSA	⑥ DSP
2	TOP,	HEX	0	27		LDA	SP I
3		IOF		28		⑦ BUN	EPI I
4		SKI		29	DSP,	HEX	0
5		BUN	OUT	30		LDA	SP
6		BUN	INP	31		⑧ ADD	MIN
7	PRO,	HEX	0	32		STA	SP
8		STA	SP I	33		BUN	DSP I
9		① ISZ	SP	34	MIN	DEC	⑨ -1
10		② CIR		35	INP,	BSA	PRO
11		STA	SP I	36		~~OUT~~ INP	
12		ISZ	SP	37		STA	CHI
13		③ LDA	TOP	38		BSA	⑩ EPI
14		STA	SP I	39		ION	
15		ISZ	SP	40		BUN	TOP I
16		ION		41	OUT,	BSA	PRO
17		BUN	PRO I	42		LDA	CHO
18	EPI,	HEX	0	43		⑪ OUT	
19		IOF		44		BSA	EPI
20		BSA	④ DSP	45		ION	
21		LDA	SP I	46		BUN	⑫ TOP I
22		STA	TOP	47	CHI,	HEX	0
23		BSA	DSP	48	CHO,	HEX	0
24		LDA	SP I	49		ORG	4000
25		⑤ CIL		50	SP,	DEC	4001

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~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** : ☐ **(D7'I)'** : ☐	No-operation
t_1	AR1 ← PC	AR1 ← 0, TR ← PC	No-operation	DR ← M[AR]
t_2	IR ← M[AR1], PC ← PC + 1	M[AR1] ← TR PC ← 0	No-operation	DR ← DR + 1
t_3	D0…D7 ← dec(IR(12-14)), AR1 ← IR(0-11), I ← IR(15)	PC ← PC + 1	No-operation	M[AR] ← DR, IF(DR=0) then (PC ← PC + 1)
t_4	No-operation	No-operation	No-operation	

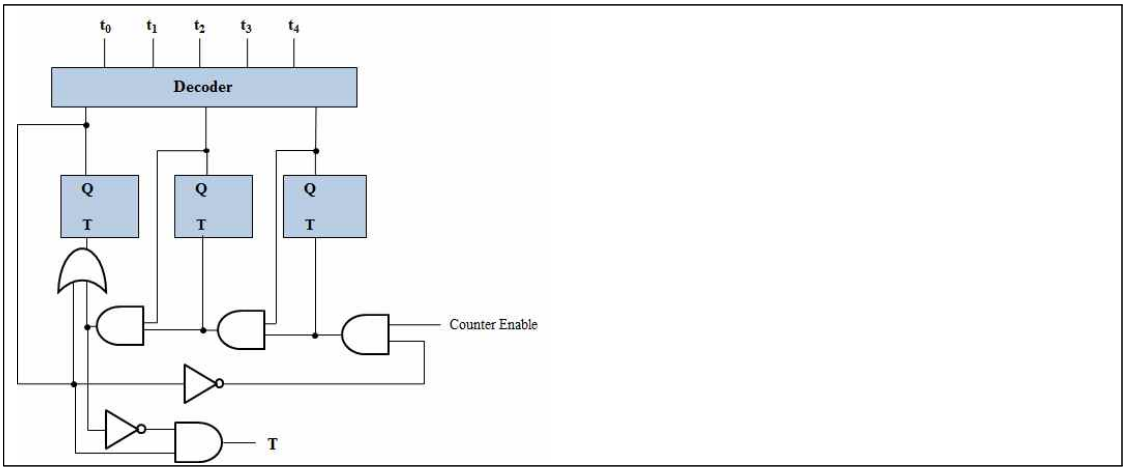
2.1 AR1 register를 따로 두어야 하는 이유를 설명하라. **[2점]**

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

2.2 IP단계의 t_0.부분의 RTL의 빈칸을 채워라. **[각2점]**

D7'I	AR← M[AR1]	**(D7'I)'**	AR← AR1

2.3 이 pipeline의 동작을 동기화하기 위한 counter register는 각 단계 내부의 time signal $t_0 \cdots t_4$를 생성하고, t_4가 끝날 때마다 생성되는 time signal은 각 단계의 동기화에 이용된다. 이를 위한 counter logic을 설계하라. **[3점]**



2.4 IF, IT, EX 단계의 t_0에 no-operation이 들어가는 이유를 설명하라. **[2점]**

Data dependency에 대한 언급이나 충돌이 발생하기 때문이라는 언급 시 정답처리

2.5 EX 단계의 t_4에 빈칸을 메워라. **[2점]**

(IEN)(FGI+FGO): R ← 1

2.6 위 Instruction Pipeline이 speedup 효과를 보려면 최소한의 프로그램 크기는 몇 개의 명령어가 되어야 하는가? **[2점]**

11개. 최소한을 가정하였으므로, T_6까지 수행되는 명령어(ISZ)만으로 명령어 구성을 가정. 11개의 명령어 수행 시, pipeline이 없는 5장의 mano computer에서는 66 clocks, 위 pipeline을 사용할 경우에는 65 clocks 동안 수행된다.

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

(1) Addressing을 하지 않는다면, assembler는 single pass로 가능하다.

(2) Postfix표현의 수식은 stack이 없으면 연산이 불가능하다.

(3) Floating point 숫자의 덧셈연산을 pipeline으로 할 때, normalization이 필요한 이유는 mantissa field의 범위가 0.5<=mantissa<1을 유지하여야하기 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 메모리를 access할 수 있다.

(7) Floating point 연산을 arithmetic pipeline으로 병렬 처리하는 경우, 처리해야할 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) Pipeline의 speedup 효과가 이상적인 결과를 낼 수 없는 이유는 각 segment를 수행하는 데 걸리는 시간이 동일하지 않기 때문이다.

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

4. 다음 성질들을 RISC와 CISC로 구분하여 R과 C로 각각 기입하라. [각2점]

Characteristic	C or R
대부분의 명령어들의 사용빈도가 유사하다.	R
다양한 Addressing Mode를 채택하고 있다	C
Operand가 메모리인 명령어들을 많이 가진다.	C
매 사이클마다 하나의 명령어를 수행할 수 있다.	R
Controller는 Logic circuit로 구현한다.	R
프로그램의 성능은 컴파일러의 성능에 좌우된다.	R
파라미터 전달을 두 모듈 간의 공유 레지스터로 수행된다.	R
명령어 파이프라인을 사용할 수 있다.	R
상대적으로 많은 레지스터를 확보하고 있다.	R
명령어의 operand를 다양하게 가져간다.	C

5. 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)

5.1. DSZ instruction을 위한 RTL을 작성하라. **[5점]**

D_6T_4: TR ← AC	D_6T_4: DR ← M[AR]
D_6T_5: DR ← M[AR], AC ← 0	D_6T_5: AC ← DR, DR ← AC
D_6T_6: AC ← complement AC	D_6T_6: AC ← complement AC
D_6T_7: AC ← AC + DR	D_6T_7: AC ← AC + 1
D_6T_8: M[AR] ← AC, if (AC = 0) then (PC ← PC + 1)	D_6T_8: AC ← complement AC
D_6T_9: DR ← TR	D_6T_9: M[AR] ← AC, if (AC = 0) then (PC ← PC + 1)
D_6T_{10}: AC ← DR, SC ← 0	D_6T_{10}: AC ← DR, SC ← 0

5-2. DSZ instruction이 없는 경우 이를 대신할 프로그램 모듈을 작성하라.[6점]

Line	Subroutine call 방법		
1		BSA	DSZ
2		HEX	(EA)

※ EA: Effective Address

Line	Subroutine		
1	DSZ	HEX	0
2		STA	TMP
3		LDA	DSZ I
4		ISZ	DSZ
5		STA	PTR
6		LDA	PTR I

Line	Subroutine		
7		ADD	MIN
8		STA	PTR I
9		SZA	
10		BUN	NSK
11		ISZ	DSZ
12	NSK	LDA	TMP
13		BUN	DSZ I
14	MIN	DEC	-1
15	PTR	HEC	0
16	TMP	HEX	0

6. 다음 질문에 답하라.

LINE	프로그램			LINE	프로그램		
1		ORG	3000	21		STA	Y
2		LDA	N	22		SZE	
3		BSA	MUL	23		BUN	ONE
4	FST,	DEC	0003	24		BUN	ZRO
5	SND,	DEC	0005	25	ONE,	LDA	X
6	TRD,	HEX	0	26		ADD	P
7	END	HLT		27		STA	P
8	N,	DEC	-2	28		CLE	
9		ORG	2000	29	ZRO,	LDA	X
10	MUL,	HEX	0	30		CIL	
11		STA	CTR	31		STA	X
12		LDA	MUL I	32		ISZ	CTR
13		STA	X	33		BUN	LOP
14		ISZ	MUL	34		LDA	P
15		LDA	MUL I	35		STA	MUL I
16		STA	Y	36		ISZ	MUL
17		ISZ	MUL	37		BUN	MUL I
18	LOP,	CLE		38	CTR,	DEC	0
19		LDA	Y	39	X,	HEX	0
20		CIR		40	Y,	HEX	0
				41	P.	HEX	0

6.1 두 변수 X. Y가 프로그램이 실행되는 과정에 변화되는 값을 작성하라. [각1점]

X	0	->	3	->	6	->	12
Y	0	->	5	->	2	->	1

6.2 서브프로그램이 시작될 때, 메인프로그램에서 무엇을 통하여 데이터를 전달하고, 그 값은 얼마인가? **[각1점]**

방법	AC	E
값	-2	0 or 1 (unknown)

6.3 최종 결과가 저장되는 장소의 주소는 얼마이고, 그 결과 값은 얼마인가? **[2+4점]**

주소	3004
결과값	3

6.4 35번 LINE을 삭제하고, 6번과 7번 LINE 사이에 새로운 명령어를 추가하여 이전과 동일한 결과를 얻는 프로그램으로 수정하여라. **[2점]**

LINE	프로그램		
7		STA	TRD
7.5	END	HLT	

6.5 이 프로그램이 수행되는 과정에서 오버플로우가 발생하는 경우를 처리하기 위하여 26번과 27번 사이에 다음 명령어를 끼워넣었다. 서브루틴 ERR의 빈곳을 메워라. (ERR은 오버플로우가 발생할 경우 프로그램을 즉시 종료시키며, 4번과 5번 LINE의 데이터는 항상 양수라고 가정한다) **[각2점]**

26~27		BSA	ERR
42		ORG	3500
43	ERR	HEX	0
44		① SNA	
45		BUN	② ERR I
46		BUN	END