

2과목 : 전자계산기구조

Check 1. 논리회로 개념

[출제빈도: 下]

1. 논리회로 정의

-2진 정보(1,0)를 기반으로 AND, OR, NOT 논리연산에 따라 수행하는 논리소자들로 구성오인 전자회로

* 논리회로의 집합(CPU) -> 전기신호

1 : 높은 전압 (5V)
0 : 낮은 전압 (1.5V)



* 'A' 가 입력되면 컴퓨터는?

1 0 0 0 0 0 1



* 1+2 가 처리되는과정?

(入) 입력장치로 데이터를 입력

(M) 메모리에 저장

(CPU) CPU 에서 호출

(CPU) CPU 에서 해독

(CPU) CPU 에서 연산

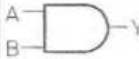



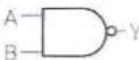



(出) 출력장치로 정보가 보여짐

* 생산적인 논리회로 설계?

-> 논리회로 간소화 -> 가격 대비 성능 극대화

2. 논리게이트 (논리소자) ★★★★★

-논리회로를 구성하는 기본 소자

게이트	기호	의미	진리표	논리식															
AND		입력신호가 모두 1일 때 1 출력	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1	$Y = A \cdot B$ $Y = AB$
A	B	Y																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
OR		입력신호 중 1개만 1이어 도 1 출력	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	1	$Y = A + B$
A	B	Y																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	1																	
NOT		입력된 정보 를 반대로 변 환하여 출력	<table><tr><th>A</th><th>Y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	Y	0	1	1	0	$Y = A'$ $Y = \bar{A}$									
A	Y																		
0	1																		
1	0																		
BUFFER		입력된 정보 를 그대로 출력	<table><tr><th>A</th><th>Y</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	A	Y	0	0	1	1	$Y = A$									
A	Y																		
0	0																		
1	1																		
NAND		NOT + AND, 즉 AND의 부 정	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	1	0	1	1	1	0	1	1	1	0	$Y = \overline{A \cdot B}$ $Y = \overline{AB}$
A	B	Y																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	
NOR		NOT + OR, 즉 OR의 부 정	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	0	$Y = \overline{A + B}$
A	B	Y																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	
XOR		입력되는 것 이 모두 같으 면 0, 한 개 라도 틀리면 1 출력	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	0	$Y = A \oplus B$ $Y = \bar{A}B + A\bar{B}$
A	B	Y																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	
XNOR		NOT + XOR, 즉 XOR의 부 정	<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	1	$Y = A \odot B$ $Y = \overline{A \oplus B}$ $Y = AB + \bar{A}\bar{B}$
A	B	Y																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	1																	

3. 생활 속의 논리 회로 - 예

*1층에서 100층까지 이동하는 엘리베이터에 내장될 전자장치를 설계하시오.

문: 열림(0), 닫힘(1) +버튼: 누르지 않은 상태(0), 누름(1)
->작동여부 결정 (0,1)

Check 2. 불대수

[출제빈도: 下]

1. 불대수

-논리회로 간소화를 위해 이용하는 논리식

2. 불대수 기본공식 ★★★★★

일반법칙	$A+A=A$, $AA=A$, $A+A'=1$, $AA'=0$ $1+A=1$, $1A=A$, $0A=0$ (A')' = A
교환법칙	$A+B = B+A$
분배법칙 /결합법칙	$A+A'B=(A+A')(A+B) = 1(A+B) = A+B$
드모르간 법칙	$(A+B)' = A' \cdot B'$, $(A \cdot B)' = A' + B'$

A	B	(A+B)	(A+B)'
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

A	B	A'	B'	A' · B'
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

* $A+B : A \cup B$

$AB : A \cap B$

$A' : A^c$

0 : 공집합(\emptyset)

1 : 전체집합(U)

3. 카르노 맵 ★★★★★

1) 간략화 방법

① 1이라고 표시된 부분을 묶는다(2^n 개씩)→중복가능

② 묶음은 곱으로, 묶음과 묶음은 합으로

③ 입력이 2개

A \ B	0	1
0	0	1
1	1	1

$A' \cdot B' + A \cdot B' + AB \rightarrow B' + A$

④ 입력이 3개

A \ BC	00	01	11	10
0	0	1	0	1
1	1	1	1	1

$$A'B'C' + AB'C' + AB'C + ABC + ABC' + A'BC' \rightarrow C' + A$$

4. 논리회로 설계 단계 ★★★★★

- (1) 요구사항 분석(문제점)
- (2) 진리표 작성
- (3) 논리식 작성
- (4) 간소화 (불대수, 카르노 맵)
- (5) 논리회로도 작성
- (6) 논리회로 구현

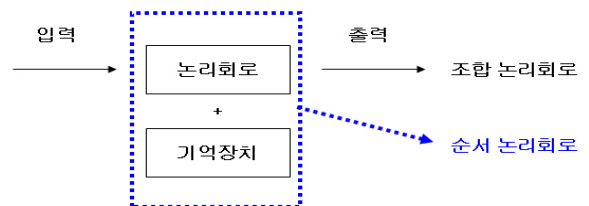
Check 3. 논리회로

[출제빈도: 下]

1. 논리회로 분류 ★★★★★

조합 논리회로	순서 논리회로
-기억능력 없음 -입력신호에 의해서만 출력 결정 -gate 집합 ex> 반가산기, 전가산기 디코더, 인코더, 멀티플렉서, 디멀티플렉서	-기억능력 있음 -입력신호와 현재신호에 의해 출력 결정 -gate + Filp Fiop 집합 ex> 카운터

*Filp Fiop(f/f): 1 bit를 기억할 수 있는 기억장치



2. 반가산기(Half Adder) ★★★★★

-2진수 1자리의 덧셈기

1) 진리표

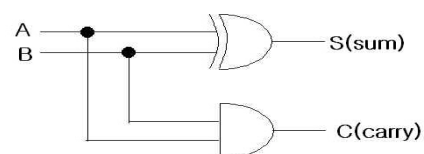
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2) 논리식

$$S = A' \cdot B + A \cdot B' = A \oplus B$$

$$C = A \cdot B$$

3) 논리회로



3. 전가산기(Full Adder) ★★★★★

-자리올림을 포함시켜 1비트 크기의 2진수를 더해서 합과

자리올림 구하는 논리회로

1) 진리표

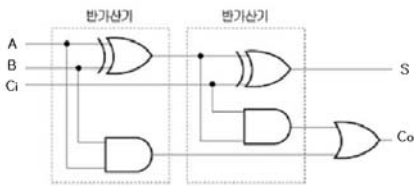
A	B	C _i	S	C _o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

2) 논리식

$$S = A' \cdot B' \cdot C_i + A' \cdot B \cdot C_i' + A \cdot B' \cdot C_i' + A \cdot B \cdot C_i \rightarrow A \oplus B \oplus C_i$$

$$C_o = A' \cdot B \cdot C_i + A \cdot B' \cdot C_i + A \cdot B \cdot C_i' + A \cdot B \cdot C_i \rightarrow (A \oplus B)C_i + AB$$

3) 논리회로

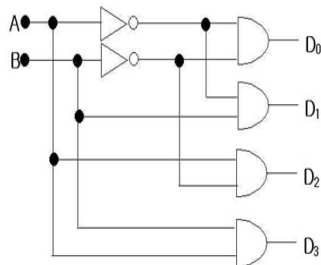


4. 디코더(Decoder, 해독기) ★★★★★

- 1) 암호형태로 전달된 정보를 원래대로 복원
<人 (암호) → 기계어(원신호)>
- 2) n개의 입력선, 2ⁿ개의 출력선
- 3) AND gate로 구성

(2X4 디코더)

A	B	D0	D1	D2	D3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



5. 인코더(Encoder, 부호기) ★★★★★

- 1) 어떤 특정한 장치에서 사용되는 정보를 다른 곳으로 전송하기 위해 일정한 규칙을 따라 암호로 변환
- 2) 2ⁿ개의 입력선, n개의 출력
- 3) OR gate로 구성

6. 멀티플렉서(MUX, 선택기) ★★★★★

- 1) 2ⁿ개의 입력선 중에서 하나를 선택하여 출력선을 전달하는 회로 (2ⁿ→1)
- 2) OR gate로 구성

7. 디멀티플렉서(DeMUX) ★★★★★

- 1) 1개의 입력신호를 가지며, 2ⁿ개의 출력선으로 구성된 회로
- 2) AND gate로 구성

8. 순서 논리회로 (gate+Flip Flop) ★★★★★

*플립플롭(f/f)

- 전원이 공급되고 있는 한, 상태의 변화를 위한 신호가 발생할 때까지 현재의 상태를 그대로 유지하는 논리회로
- 레지스터를 구성하는 기본소자
- 반도체 기억장치에서 2진수 1자리 값을 기억하는 메모리 소자

f/f	특성표				
RS	S	R	Q(t)	Q(t+1)	불변 reset set 불허
	0	0	0,1	Q(t)	
	0	1	0,1	0	
	1	0	0,1	1	
	1	1	0,1	X	
JK	-RS f/f 불허 조건을 해결				
	J	K	Q(t)	Q(t+1)	불변 reset set 보수
	0	0	0,1	Q(t)	
	0	1	0,1	0	
	1	0	0,1	1	
1	1	0,1	Q(t)'		
D	-RS f/f에서 입력이 배타적				
	D	Q(t)	Q(t+1)		
	0	0,1	0		
	1	0,1	1		
T	-JK f/f에서 입력이 동일 클릭펄스가 가해질 때마다 출력상태가 반전				
	T	Q(t)	Q(t+1)		
	0	0,1	Q(t)		
	1	0,1	Q(t)'		

Check 4. 자료의 개념

[출제빈도: 下]

1. 자료의 단위 ★★★★★

비트(bit)	-정보(자료) 표현의 최소 단위 - 두 가지 상태(0과1)를 표시하는 진수 1자리 *2bit= $2^2=4$ 개 *3bit= $2^3=8$ 개
니블(Nibble)	-4개의 비트가 모여 1개의 니블을 구성 -4비트로 구성되며 16진수 1자리를 표현하기 적합
바이트(Byte)	-문자를 표현하는 최소 단위로 8개의 비트가 모여 1바이트를 구성함 -주소 지정의 단위로 사용
워드(word)	-컴퓨터가 한번 처리할 수 있는 명령단위 -Half Word = 2byte -Full Word = 4byte -Double Word = 8byte

Check 5. 진법

[출제빈도: 下]

1. 진법변환 ★★★★★

1) 진법

-2진수(0,1), 8진수(0~7), 16진수(0~9,A~F)

2) 진법 변환

① 10진수→N진수 ★★★★★

:10진수를 N으로 나누어서 나머지를 거꾸로 처리

예) (16)₁₀ → (10000)₂

2	16	
2	8	0
2	4	0
2	2	0
1	1	0

기출) 십진수 21.6 → 2진수 10101.1001

② N진수→ 10진수 ★★★★★

: 각 자리의 가중치를 계산

예) (10000)₂ → (16)₁₀

1	0	0	0	0
2^4	2^3	2^2	2^1	2^0

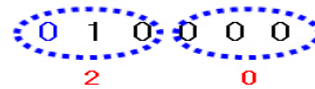
$$\rightarrow (1 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = (16)_{10}$$

기출) (101110.1101)₂ → 10진수 46.8125

기출) 8진수 23.32 → 10진수 19.406

③ 2진수 ↔ 8진수 (3자리씩 묶어서 처리)

예) (10000)₂ ↔ (20)₈



④ 2진수 ↔ 16진수 (4자리씩 묶어서 처리)

예) (10000)₂ ↔ (10)₁₆



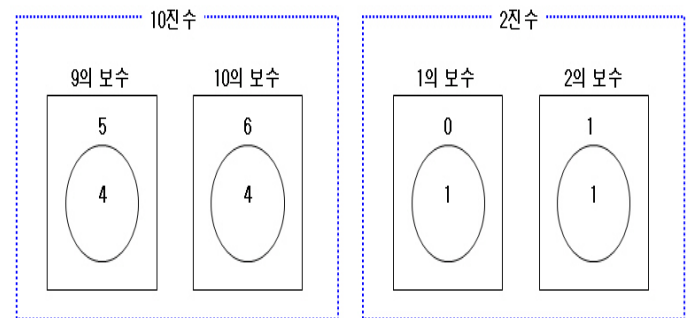
⑤ 8진수 ↔ 16진수 (2진수로 변환 후 처리) ★★★★★

Check 6. 보수

[출제빈도: 下]

1. 보수 (Complement) ★★★★★

: 서로 상반되는 수 (보수를 이용하면 가산기(덧셈)을 이용하여 뺄셈을 할 수 있다.)



→ 10의 보수, 2의 보수는 각각 9의 보수와 1의 보수에서 1을 더하면 된다.

기출) 10진수 274 → 9의 보수 725

기출) 이진수 011001 → 1의 보수 100110

기출) 이진수 1001011 → 2의 보수 0110101

기출) 10진수 5 → 4자리 1의 보수 1010, 2의 보수 1011

2. 음수표현 방법 ★★★★★

: 컴퓨터에서 음수를 표현하는 3가지 방법 (양수는 1가지 방법)

예) -14를 8bit로 표현하면

1) 부호화 절대치: 1000 1110 → 첫 비트는 부호비트이며 음수이므로 1값을 줌

2) 부호화 1의 보수: 1111 0001 → 부호비트는 고정하고 각 자리값을 바꿔서 1의 보수를 구함

3) 부호화 2의 보수: 1111 0010 → 1의 보수에서 1을 더해서 2의 보수를 구함이다.

(수 표현 범위) (8bit 일 때)

-부호화 절대치 } $\rightarrow +0, -0 \rightarrow -2^{n-1} + 1 \sim 2^{n-1} - 1 \rightarrow -2^7 + 1 \sim 2^7 - 1$

-1의 보수 $\rightarrow +0 \rightarrow -2^{n-1} \sim 2^{n-1} - 1 \rightarrow -2^7 \sim -(2^7 - 1)$

-2의 보수 $\rightarrow +0 \rightarrow -2^{n-1} \sim 2^{n-1} - 1 \rightarrow -2^7 \sim -(2^7 - 1)$

* 2의 보수 표현의 장점

- 2의 보수에서는 carry가 발생하면 버린다. (1의 보수는 더함)
- 수치를 표현하는데 있어서 0의 판단이 가장 쉬운 방법
- 표현할 수 있는 수의 개수가 하나 더 많다.

기출) $(-17) + (-4) \rightarrow$ 2의 보수 11101011

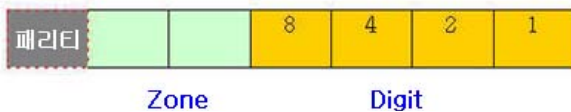
Check 7. 자료의 표현방식 -외부적 [출제빈도: 下]

1. 외부적 표현 형식

- code로 표시하여 사람이 이해할 수 있도록 표현
- 종류: BCD코드, EBCDIC코드, ASCII코드, 그레이코드, 해밍코드, 3초과 코드, 7421코드

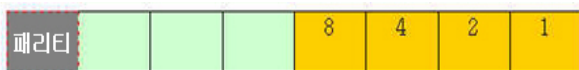
2. 기본코드

- 1) BCD code : IBM, 6bit($2^2 = 64$ 개 자료 표현), 영소문자 사용 X, 수치 계산용



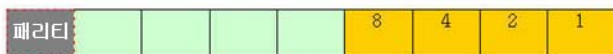
2) ASCII code ★★★★★

- : 미국 표준협회, 7bit($2^7 = 128$ 개), 통신제어용, 마이크로컴퓨터 기본코드로 사용



3) EBCDIC ★★★★★

- : IBM, BCD code 확장, 8bit($2^8 = 256$ 개), 중대형 컴퓨터에 사용



기출) 10진 숫자 5 \rightarrow EBCDIC 11110101

3. BCD 코드 (=2진화 10진수, 8421 코드) ★★★★★

- 1) 10진수 1자리를 2진수 4자리(bit)로 표현하는 가중치 코드
- 2) 10진수 입/출력이 편함

*기출) 10진수 46 \rightarrow BCD 코드 01000110

4. 3초과 코드 (Excess-3) ★★★★★

- 1) 8421 코드 + (3)₁₀
- 2) 비가중치(unweighted code), 자기보수 코드
- 3) 3초과 코드는 8421 코드와 비교해서 0000~0010을 표현할 수 없고, 추가로 1010~1100을 표현할 수 있다.

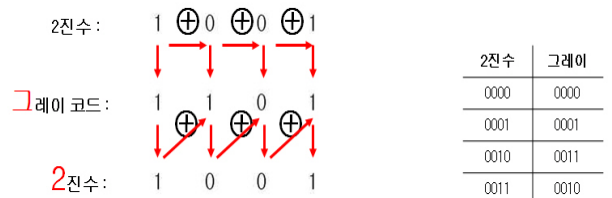
5. 해밍코드 ★★★★★

- 1) 오류검출, 정정 가능

10진수	8421	3초과
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100
10	0001 0000	0100 0011
11	0001 0001	0100 0100
12	0001 0010	0100 0101
13	0001 0011	0100 0110
23	0010 0011	0101 0110

6. 그레이 코드 (Gray code) ★★★★★

- 1) BCD 코드의 인접한 자리를 XOR 연산으로 만든 코드
- 2) 이웃하는 코드가 한 비트만 다르기 때문에 코드 변환이 용이해서 A/D 변환에 주로 사용
- 3) 입출력 장치, H/W 에러를 최소화



7. 패리티 검사 ★★★★★



\rightarrow 오류검출(O), 정정(X)

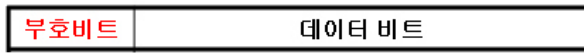
Check 8. 자료의 표현방식 -내부적 [출제빈도: 下]

1. 내부적 표현 방식 - 고정 소수점 표현 ★★★★★

- 정수 데이터의 표현 및 연산에 사용하는 방법
- 종류: 2진 표현(부호화 절대치, 부호화1의보수, 부호화2의보수) 10진 표현 (언팩: 존형식, 팩형식)

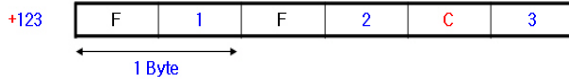
1) 2진 표현

- 표현방식: 부호비트(0이면 양수, 1이면 음수)

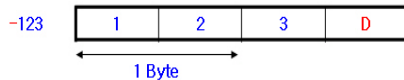


2) 10진 표현 ★★★★★

① UnPack 형식 (Zone 형식): 10진수 읽.출력



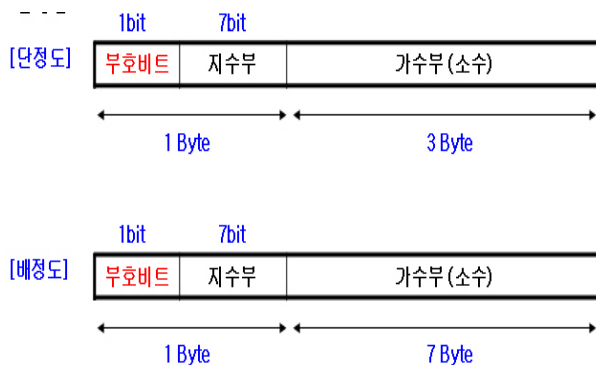
② Pack 형식: 10진수 연산



2. 내부적 표현 방식 - 부동 소수점 표현 ★★★★★

- 실수 데이터의 표현과 연산에 사용되는 방법(단정도, 배정도)
- 고정 소수점 표현보다 표현의 정밀도를 높일 수 있다.
(아주 큰수, 아주 작은 수 표현 가능)
- 과학이나 공학 또는 수학적인 응용에 주로 사용
- 고정 소수점 표현에 비해 연산이 복잡하고 연산 시간 ↑
- 자수부와 가수부를 분리하는 정규화 과정 필요

1) 표현 방법



기출) 십진수 +14925를 단정도 부동 소수점 표현하기 → 지수부=16진수 44, 소수부=3A4D

- 16진수 변환: +3A4D
- 정규화 (소수점 첫째 자리로 유효 숫자를 이동하여 가수부와 지수부를 분리): $+0.3A4D \times 16^4$
- 지수부를 64비트 이진수로 표현해서 100 0000 + 100 = 100 0100 → 지수부는 16진수 44 저장
- 소수부는 그대로

$$+0.3A4D \times 16^4$$

0	100 0100	0011 1010 0100 1101 0000 0000
---	----------	-------------------------------

+ 44 3 A 4 D

* 7비트로 표현할 수 있는 지수부 범위: -64 ~ 0 ~ 63

2) 부동 소수점 수 연산법 ★★★★★

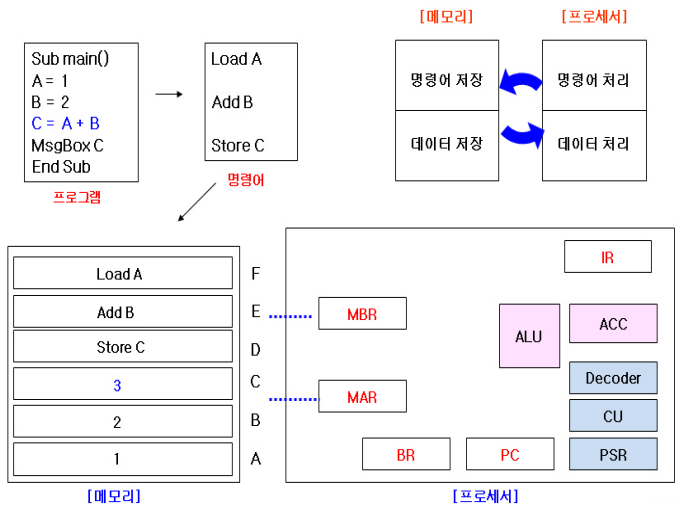
- 가감산: 두수의 자수부가 같도록 소수점의 위치를 조정
(지수가 큰 쪽에 맞춘다.)
- 승제산: 지수부와 가수부를 별도로 처리

(승산: 지수부 +, 가수부 X)

Check 9. 중앙처리장치

[출제빈도: 上]

* 개념 이해하기



1. 중앙처리장치 (CPU) 구성 ★

1) 제어장치 (CU : Control Unit)

: 명령을 꺼내서 해독하고, 시스템 전체에 지시 신호를 내는 것 (제어기능)

- Decoder : 명령레지스터에 호출된 OP Code를 해독하여 그 명령을 수행시키는데 필요한 각종 제어신호를 만들어 내는 장치
- 순서기, 주소처리기

2) 연산장치(ALU : Arithmetic Logic Unit)

: 실제 연산하는 장치 (연산기능)

3) 레지스터 : CPU 속에서 일시적으로 값을 기억하는 임시기억장소 (기억기능)

- PC (Program Counter) : 다음에 실행할 명령의 번지 기억 (Next Instruction Address)
- IR (Instruction Register) : 현재 수행 중인 명령의 내용 기억
- ACC (Accumulator 누산기) : 연산의 결과를 일시적으로 저장
- MAR (Memory Address Register) : 데이터의 번지를 저장
- MBR (Memory Buffer Register) : 기억장치에서 참조한 데이터를 저장
- SR (Program Status Register, Major state register) : 컴퓨터 상태를 나타내는 레지스터
- * PSW (Program Status Word) : 시스템 순간 순간 상태에 대한 정보 (레지스터 X)
- FR (Flag Register) : 레지스터 가운데 명령어를 수행 할 때마다 결과가 0인지 여부, 부호(음수 인지 양수인지), 캐리 및 오버플로의

발생 여부 등을 각각 1비트로 나타내는 레지스터

4) 버스 : 장치들 간 상호 필요한 정보를 교환하기 위해 연결하는 공동의 전송선 (전달기능)

- Address Bus, Data Bus (양방향성), Control Bus

Check 10. 명령어

[출제빈도: 上]

1. 명령어 (Instruction) 구성 ★

1) 연산자부 (Operation code, OP code)

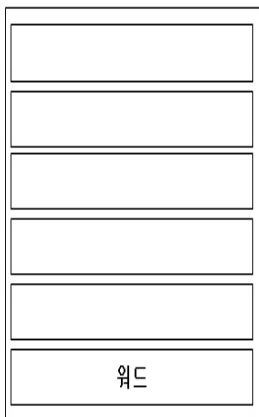
- 수행해야 할 동작에 맞는 연산자
- 크기(bit)는 표현할 수 있는 명령 개수 (2^n 개)
- 모드(mode) 비트: 직접 주소(0), 간접 주소(1)

연산자부	주소부
------	-----

예) Load A

2) 주소부 (Operand, OP)

- 기억장소의 주소, 레지스터 번호, 사용할 데이터
- 크기는 메모리 용량과 관계



OP code: 4bit

mode bit: 1bit

메모리 용량: 1,024워드

OP: ?

명령의 크기: ?

* Word: 컴퓨터가 한번에 처리할 수 있는 명령의 단위

2. 연산자 (OP code) 기능 (= 명령어 기능) ★

함수연산 기능(처리)	- 산술연산 +, -, x, ÷, 산술 shift - 논리연산 AND, OR, NOT, XOR 논리 shift - ADD, CPA, CLC (Clear Carry 명령) ROL (Rotate Left 명령) ROR
자료전달 기능	- Load (M/M ← CPU) - Push, Pop, Move
제어 기능	- 프로그래머가 명령의 실행 순서를 제어 - 분기명령 (Branch) - Call, Return, JMP (jump 명령), SMA
입/출력 기능	- CPU ↔ I/O 장치, 메모리 ↔ I/O 장치 - INT, OUT

3. 주소지정방식 (OP 개수에 따라) ★

1) 3주소 명령어

OP Code	OP 1	OP 2	OP 3
---------	------	------	------

예) Add A B C

연산의 결과 저장

- 장점: 원래 자료 유지, 프로그램 전체 길이 짧게, 주기억장치 접근횟수 줄어든다.
- 단점: 명령어 1개 길이가 길다. 수행시간 길다

2) 2주소 명령어

OP Code	OP 1	OP 2
---------	------	------

예) Add A B

연산의 결과 저장

- 장점: 3주소보다 명령어 길이 짧다.
- 단점: 전체 프로그램 길이가 길어진다. OP1의 값이 소멸

3) 1주소 명령어

OP Code	OP 1
---------	------

예) Add A

- 누산기(ACC)를 이용

4) 0주소 명령어

- Stack 이용

4. 주소지정방식 (OP 개수에 따라) ★

즉시 주소 지정 (Immediate)	- 오퍼랜드(operand) 부분에 데이터를 기억, 속도가 가장 빠르다. 데이터 값 범위 제한
직접 주소 지정 (Direct)	- 명령의 주소부가 사용할 자료의 번지를 직접 표현
간접 주소 지정 (Indirect)	- 명령문 내의 번지는 실제 데이터의 위치를 찾을 수 있는 번지가 들어 있는 장소를 표시 - 인스트럭션의 길이가 짧고 제한되어 있어도 이것을 이용하여 긴 주소를 찾아 갈 수 있다.

주소	Data
:	
120	200
:	
200	300
:	
300	500
:	

• 즉시주소 지정 ⇒ 120

Load	120
------	-----

• 직접주소 지정 ⇒ 200

Load	120
------	-----

• 간접주소 지정 ⇒ 300

Load	120
------	-----

5. 계산에 의한 주소 지정 ★

상대 주소 지정 (Relative)	- 기억장소의 위치 = 명령어 주소부에 있는 주소값 + PC(Program Counter)
인덱스 주소 지정 (Index)	- 기억장소의 위치 = 명령어 주소부에 있는 주소값 + IR(Index Register)
베이스 레지스터 주소 지정 (Base)	- 기억장소의 위치 = 명령어 주소부에 있는 주소값 + BR(Base Register) - 프로그램의 재배치가 용이하다. - 다중 프로그래밍 기법에 많이 사용된다.

Check 11. 연산의 분류

[출제빈도: 中]

1. 연산자 분류 ★★★★★

1) 성질에 따른 분류

- 비수치적 연산 (논리연산) : AND, OR, XOR, NOT(Complement), 논리 Shift, Rotate, Move 등
- 수치적 연산 (산술연산) : +, -, *, /, 산술 Shift 등

2) 항에 따른 분류

- 단항(Unary) : 논리 Shift, 산술 Shift, Rotate, Not(Complement, 보수) 등
- 이항(Binary) : 사칙연산, AND, OR, XOR 등

2. AND 연산 (Masking 연산) ★★★★★

- 특정문자, 비트를 삭제
- 삭제할 부분 '0' bit (Mask bit)

A	B
11000001	11000010
11111111	00000000
11000001	00000000
A	

3. OR 연산 ★★★★★

- 특정문자를 삽입
- 특정 비트에 1을 세트(Selective-set) 시키는 연산

A	B
11000001	00000000
00000000	11000010
11000001	11000010
A	B

4. XOR 연산 ★★★★★

- 2개 데이터를 비교(compare)
- 특정비트반전

5. 논리 Shift 연산 ★★★★★

A	11000001	00001111
A	11000001	11111111
	00000000	11110000

- 왼쪽 또는 오른쪽 n bit씩 자리를 이동
- 데이터의 직렬전송
- 삽입되는 자리는 0
- 자리 범위를 넘어 서는 것은 사라진다.

← 1 bit

	1	1	1	0	0	1
1	1	1	0	0	1	0

6. 산술 Shift 연산 ★★★★★

- 부호를 고려하여 자리를 이동시키는 연산
- 2^n 곱, 2^n 나눌 때

예) (1)₁₀ 왼쪽 Shift

2^3	2^2	2^1	2^0	
0	0	0	1	= 1
0	0	0	1	0 = 2
0	0	1	0	0 = 4

예) (8)₁₀ 오른쪽 Shift

2^3	2^2	2^1	2^0	
1	0	0	0	= 8
0	1	0	0	= 4
0	0	1	0	= 2

예) (+27)₁₀ 오른쪽 Shift → 양의 홀수일 때 (0.5가 줄어든 결과)

	0	0	0	1	1	0	1	1	
	0	0	0	0	1	1	0	1	1 = (13) ₁₀

예) (-27)₁₀ 오른쪽 Shift → 음의 홀수일 때 (0.5가 늘어난 결과)

	1	0	0	1	1	0	1	1	
고정↓	1	0	0	0	1	1	0	1	1 = (-13) ₁₀

* Padding : 이동 시 삽입되는 값은? 빈 공간이 생기는데 이 빈 공간을 채우는 것

1) 왼쪽 산술 Shift (곱셈)

종류	패딩 비트
부호화 절대치	항상 0
1의 보수	양수 : 0, 음수 : 1
2의 보수	항상 0

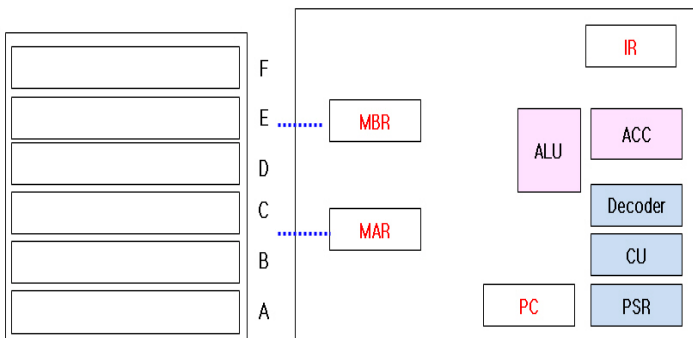
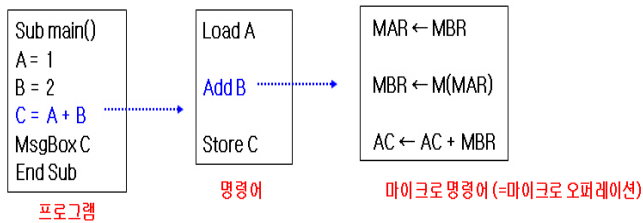
2) 오른쪽 산술 Shift (나눗셈)

종류	패딩 비트
부호화 절대치	항상 0
1의 보수	양수 : 0, 음수 : 1
2의 보수	양수 : 0, 음수 : 1

Check 11. 마이크로 오퍼레이션

[출제빈도: 中]

*개념 이해하기



1. 마이크로 오퍼레이션의 정의 ★★★★★

- 명령을 수행하기 위해 CPU내의 레지스터와 플래그가 의미 있는 상태 변환을 할 수 있도록 하는 동작
- 레지스터에 저장된 데이터의 의해서 이루어지는 동작
- 한 개의 클럭 펄스 동안 동작
- 제어신호에 의해 micro-operation이 순서적으로 일어남
- 하나의 클럭 펄스 동안에 실행되는 기본적인 동작을 의미

2. 마이크로 오퍼레이션의 종류 ★★★★★

- 마이크로 사이클 타임: 한 개의 마이크로 오퍼레이션을 수행하는데 걸리는 시간

동기 고정식	가장 긴 시가 -장점: 수행시간 비슷, 제어기 구현 단순 -단점: CPU 시간 낭비 심함
동기 가변식	수행시간의 편차가 클 경우 →수행시간이 비슷한 마이크로 오퍼레이션을 그룹화 -장점: CPU이용 효율이 좋다 -단점: 제어기 구현 복잡
비동기식	모든 마이크로 오퍼레이션 →서로 다르게 정의 -장점: CPU 시간 낭비 없다 -단점: 현실적으로 구현이 어려움

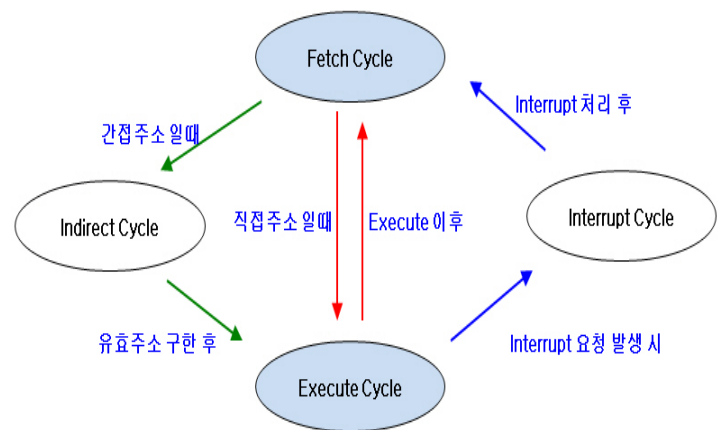
Check 12. 메이저 스테이트

[출제빈도: 中]

1. Major State 개념: CPU의 4가지 동작(상태) ★★★★★

*CPU의 명령어 수행 순서

Instruction Fetch→디코딩→Operand Fetch→Execute



- Fetch: 기억장치에서 Instruction을 읽어서 CPU로 가져옴
→IR(명령레지스터)저장→해독
- Indirect: Fetch 단계에서 해석된 명령의 주소부(operand)가 간접주소인 경우 수행됨
- Execute: Fetch 단계에서 인출하여 해석한 명령(연산)을 실행하는 단계
- Interrupt: 인터럽트 발생 시 복귀주소(PC)를 저장시키고, 제어순서를 인터럽트 처리 p/g의 첫 번째 명령으로 옮기는 단계

2. Major State (Micro Operation) ★★★★★

- * MAR←PC : 명령을 수행하는 과정에서 가장 먼저 수행되어야 하는 마이크로 오퍼레이션 ★★★★★

3. Interrupt (개념설명)

Fetch $MAR \leftarrow PC$ $MBR \leftarrow M(MAR), PC \leftarrow PC+1$ $IR \leftarrow MBR$	Indirect $MAR \leftarrow MBR$ $MBR \leftarrow M(MAR)$
Interrupt $MBR \leftarrow PC, PC=0$ $MAR \leftarrow PC, PC=PC+1$ $M(MAR) \leftarrow MBR, IEN=0$ goto Fetch	Execute ADD LDA STA ISZ

MBR :
IR :
MAR :
PC :

0	
1	
100	
101	

MBR:
MAR:
PC:

$MBR \leftarrow PC, PC=0$
 $MAR \leftarrow PC, PC=PC+1$
 $M(MAR) \leftarrow MBR, IEN=0$
 goto Fetch

* IEN => 1 : 인터럽트 수행 중
 0 : 인터럽트 수행 완료
 * 메모리 0번지에 복귀주소 기억

4. ADD ★★★★★

-AC의 내용과 메모리의 내용을 ADD→결과 AC

$MAR \leftarrow MBR$
 $MBR \leftarrow M(MAR)$
 $AC \leftarrow AC + MBR$

5. AND

-AC(누산기) 내용과 메모리 내용을 AND연산→결과 AC

$MAR \leftarrow MBR$
 $MBR \leftarrow M(MAR)$
 $AC \leftarrow AC \text{ AND } MBR$

6. LDA ★★★★★

-메모리 내용을 AC로 가져오는 것(load)

$MAR \leftarrow MBR$
 $MBR \leftarrow M(MAR), AC \leftarrow 0$
 $AC \leftarrow AC + MBR$

7. STA (store AC) ★★★★★

-AC의 내용을 메모리 저장

$MAR \leftarrow MBR$
 $MBR \leftarrow AC$
 $M(MAR) \leftarrow MBR$

8. BUN (Branch Unconditionally)

-PC에 특정한 주소를 전송하여 실행명령의 위치를 변경
 →무조건 분기 명령

$PC \leftarrow MBR[AD]$

9. ISZ (Increment and Skip if zero)

-메모리의 값을 읽어, 그 값을 1 증가

$MAR \leftarrow MBR$
 $MBR \leftarrow M(MAR)$
 $MBR \leftarrow MBR + 1$
 $M(MAR) \leftarrow MBR$

Check 13. 제어장치

[출제빈도: 上]

1. 제어장치 ★★★★★

- 1) 제어 신호를 보내는 역할
 (필요한 마이크로 연산들이 연속적으로 수행)
- 2) **제어신호**: 중앙연산처리장치에서 마이크로 동작이 순서적으로 일어나게 하는 신호
- 3) 제어 데이터: 제어 장치가 제어신호를 발생시키기 위한 데이터
 (수치 데이터)
 -메이저 스테이트 사이의 변환을 제어하는 제어 데이터
 -중앙처리장치의 제어점을 제어하는데 필요한 제어 데이터
 -인스트럭션 수행 순서를 결정하는데 필요한 제어 데이터
- 4) 제어 기억장치: ROM으로 구현

2. 제어장치의 종류 ★★★★★

- 1) 하드 와이어드(고정 배선 제어장치) : H/W
 -고속, 고가, 한번 만들어진 명령어 세트 변경 불가, 회로구성 복잡
- 2) 마이크로 프로그램(ROM): S/W
 -어떤 명령을 수행할 수 있도록 된 일련의 제어 워드가 특수한 기억 장치 속에서 저장된 것

- 저속, 저가, 명령어 세트를 쉽게 변경
- 마이크로 명령어: 한 마이크로 사이클 동안 발생해야 되는 제어신호
- ① 수평마이크로 명령: 마이크로 명령어의 한 비트가 한 개의 마이크로 동작
- ② 수직 마이크로 명령: 한 개의 마이크로 명령으로 한 개의 마이크로 동작
- ③ 나노 명령: 나노 메모리에 저장된 마이크로 명령

Check 13. 입출력장치

[출제빈도: 上]

1. 입출력 장치(INPUT/OUTPUT) ★★★★★

- 구성(기능): 입/출력 인터페이스, 입출력 제어, 입출력 버스
- 입출력 방식: DMA, 채널, 인터럽트, 프로그램(CPU)에 의한 입출력(폴링)
- 드루풋(throughput): 비율: 폴링<인터럽트<DMA<채널

2. I/O와 M/M의 차이점 ★★★★★

- 1) 동작 속도 : M/M > I/O → 입출력 제어기(Interface)가 필요한 가장 큰 이유
- 2) 정보 단위 : M/M (word) > I/O (Byte)
- 3) 오류 발생률 : M/M < I/O
- 4) 동작 주체 : M/M (CPU), I/O (독립적)

3. 채널(Channel) ★★★★★

- I/O 전용 프로세서 (입·출력장치와 주기억장치를 연결하는 중개 역할)
- CPU의 명령을 받고 입출력 조작을 개시하면 CPU와는 독립적으로 조작
- 종류
 - ① 선택채널 (Selector) : 고속 입출력 장치, 특정한 한 개의 장치를 독점
 - ② 다중채널 (Multiplexer) : 저속 입출력 장치, 동시에 여러 개의 입출력 제어
 - ③ Block Multiplexer : Selector 와 Multiplexer 방식을 결합

4. DMA (Direct Memory Access) ★★★★★

- 기억소자와 I/O 장치간의 정보교환 때 CPU의 개입 없이 직접 정보 교환이 이루어 질 수 있는 방식
- Cycle steal(DMA 제어기가 한번에 한 데이터 워드를 전송하고 버스의 제어를 CPU에게 돌려주는 방법) 이용
- DMA는 입출력 전송에 따른 CPU의 부하를 감소시킬 수 있다
- 보다 빠른 데이터의 전송이 가능
- 인터럽트와 차이점

인터럽트	사이클 스틸
CPU 상태 보존 필요	상태 보존 불필요
CPU 수행 계속	대기상태

5. 인터럽트 ★★★★★

- CPU가 직접 제어하는 방식 중에서 입/출력 장치의 요구가 있을 때 데이터를 전송하는 제어 방식

Check 14. 인터럽트

[출제빈도: 上]

1. 인터럽트 ★

- 1) 정의 : 전자계산기에서 어떤 특수한 상태 (예기치 않은 응급사태)가 발생하면 그것이 원인이 되어 현재 실행하고 있는 프로그램이 일시 중단되고, 그 특수한 상태를 처리하는 프로그램으로 옮겨져 처리한 후 다시 원래의 프로그램을 처리하는 현상
- 2) 인터럽트 요인이 받아들여 졌을 때 CPU가 확인하여야 할 사항 : PC, 상태조건, 관련(모든) 레지스터 상태 확인
- 3) 인터럽트 수행 후에 처리 할 사항 : 인터럽트 처리 시 보존시켰던 PC 및 제어상태 데이터를 PC와 제어상태 레지스터에 복구
- 4) 인터럽트 체제의 기본 요소 : 인터럽트 요청 신호, 인터럽트 취급 루틴, 인터럽트 처리 기능

2. 인터럽트 수행 순서 ★

- 1) Interrupt 요청신호 발생
- 2) 현재 수행 중인 명령을 완료하고, 상태 기억 (복귀주소 : M/M 0번지, Stack M)
- 3) Interrupt 판별
- 4) ISR 에 의해 Interrupt 처리
- 5) 보존한 프로그램 상태 복구 후 계속 처리

3. 인터럽트 종류 ★

내부(Internal) 인터럽트	외부(External) 인터럽트
-프로그램에 의한 인터럽트 =트랩(trap) -0으로 나눔, 스택의 overflow -우선순위가 낮음	-정전(Power Fail) : 우선순위가 가장 높음 -Timer에 의한 인터럽트 -입출력 인터럽트 -operator가 임의로 발생시킬 수 있음

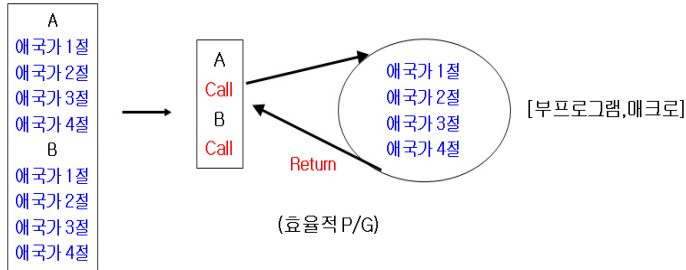
※ 참고 ★★★★★

- * 인터럽트 X : 임의의 부프로그램(서브프로그램) 호출, 분기(Branch) 명령
- * Use Bad Command Interrupt

: 정의되지 않은 명령이나 불법적인 명령을 사용했을 경우 혹은 보호되어 있는 기억공간에 접근하는 경우 발생

* 금지된 자원을 접근할 경우 S/W 문제로 프로그램에 오류가 없는데도 인터럽트가 발생한다.

4. 부프로그램(서브루틴) ★★★★★



(비효율적 P/G)

- 부프로그램과 매크로 공통점

: 여러 번 중복되는 부분을 별도로 작성하여 사용

- 매크로(MACRO)

: 프로그래머가 어셈블리 언어(Assembly language)로 프로그램을 작성할 때 반복되는 일련의 같은 연산을 효과적으로 하기 위해 필요한 것

- 리커션(recursion) 프로그램

: 한 루틴(routine)이 자기를 다시 부를 때

- Stack : 부 프로그램(Sub program)에서 주 프로그램(Main program)으로 복귀할 때 필요한 주소를 기억

5. 우선순위(priority) 판별 방법 ★

1) S/W : 폴링 (Polling)

- 인터럽트 요청신호 플래그를 차례로 검사하여 인터럽트의 원인을 판별하는 방식

2) H/W : 데이지 체인 (daisy-chain)

- 인터럽트를 발생하는 모든 장치들을 인터럽트의 우선순위에 따라 직렬로 연결함으로써 이루어지는 우선순위 인터럽트 처리방법
- 장치번호 버스를 이용한다
- 벡터(인터럽트를 발생한 장치가 프로세서에게 분기할 곳의 정보를 제공해 주는 것)에 의한 인터럽트 처리 방법
- 응답속도가 빠르다

3) 인터럽트 우선순위 체인, 인터럽트 요청 체인

6. 하드웨어와 소프트웨어의 차이점

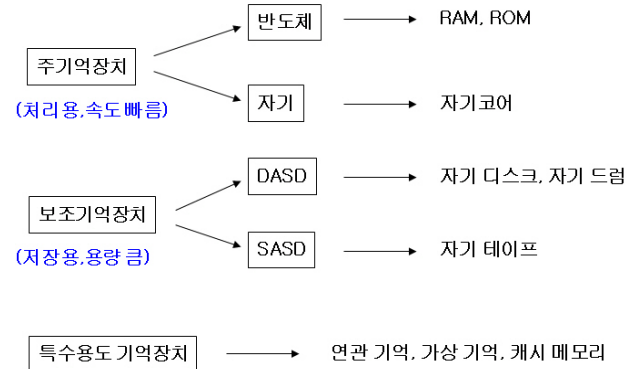
구분	H/W	S/W
반응속도	고속	저속
회로복잡도	복잡	간단
경제성	비경제적	경제적
융통성	없다	있다

Check 15. 기억장치분류

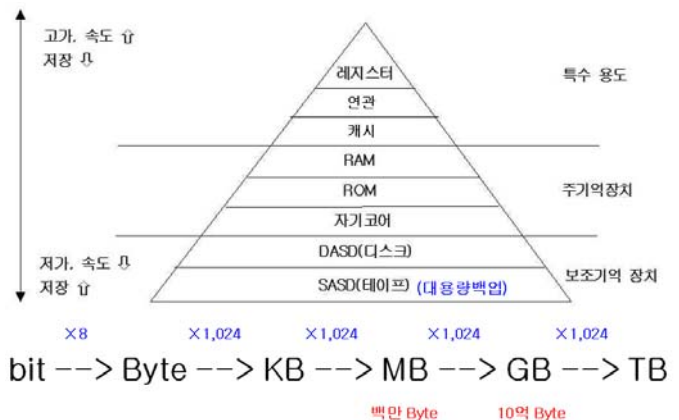
[출제빈도: 中]

1. 기억장치 분류 ★★★★★

- 1) 전원공급 유무 : 휘발성 (RAM), 비 휘발성 (ROM)
- 2) 내용 보존 유무 : 파괴 (자기코어), 비 파괴
- 3) 시간의 흐름 : 정적 (SRAM), 동적 (DRAM)-Refresh 필요
- 4) 액세스 : DASD (하드디스크-직접 접근), SASD (자기테이프-순차 접근)



2. 기억장치 계층 구조 ★★★★★



3. 기억장치 성능 평가 요소 ★★★★★

1) 기억용량, 편리성, 응답성, 신뢰도

2) 접근시간 Access Time

: 정보를 기억 장치에 기억시키거나 읽어 내는 명령을 한 후부터 실제로 정보를 기억 또는 읽기 시작할 때까지 소요되는 시간

= Seek Time + Search Time + Transmission Time

- Seek Time(탐색 시간) : 트랙을 찾는데 걸리는 시간

- Search Time(회전 지연 시간, 대기 시간)

: 섹터를 찾는데 걸리는 시간

- Transmission Time(전송 시간) : 해당 내용 전송

3) Cycle Time : 기억장치에 접근을 위하여 판독신호를 내고 나서 다음 판독신호를 낼 수 있을 때까지의 시간

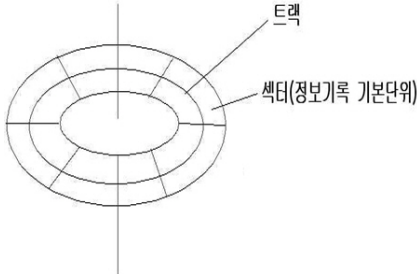
- C.T > A.T (자기코어)

4) 소요시간 Turnaround Time

: 데이터를 수집하고 그것을 계산 처리용으로 변환하여 계산을 실행 한 후 그 결과를 사용자에게 알려주는데 필요한 시간

5) Band width(대역폭, 전송률, 밴드폭)

- 기억장치 자료 처리 속도, 정보 전달능력에 한계 (Access bit 수/초당)



Check 16. 주기억장치

[출제빈도: 中]

1. 주기억장치 (Main storage) ★★★★★

- CPU가 직접 액세스 , - 현재 수행되는 프로그램, 데이터

2. ROM(Read Only Memory) ★★★★★

- Only Read, 비 휘발성, 입·출력 시스템의 자가 진단 프로그램 저장

- ① Mask ROM : 프로그램 되어 있는 ROM
- ② PROM : PROM writer로 기입되고 내용을 지울 수 없다.
- ③ EPROM : 자외선을 이용하여 지우고 다시 사용할 수 있는 메모리
- ④ EEPROM : 전기적으로 삭제하고 다시 쓸 수도 있는 기억장치

3. RAM (Random Access Memory) ★★★★★

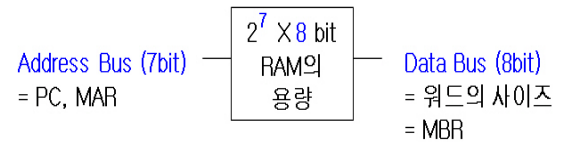
- 휘발성
- 종류: 재충전의 유무에 따라서

구 분	DRAM(동적)	SRAM(정적)
구성소자	콘덴서	플립플롭
특 성	주기적 재충전	전원공급 되는 동안만 기억 유지
전력소모	적다	많다
접근속도	느리다	빠르다
가 격	저가	고가
용 도	일반적인 주기억장치	캐시 메모리

4. 자기코어 (지움성 읽음-Destructive Read-Out) ★★★★★

- 데이터를 읽으면 읽은 내용이 지워지는 파괴 메모리(destructive memory) 이므로 내용을 읽은 후 지워진 내용을 기록하기 위한 재 저장 시간이 필요함

5. 주기억장치 용량 계산 ★★★★★



기출) 기억장치의 총 용량이 4096비트이고 워드 길이가 16비트일 때 프로그램 카운터(PC) → 8, 주소 레지스터(AR) → 8, 데이터 레지스터(DR)의 크기 → 16

기출) 컴퓨터의 메모리 용량이 16K X 32bit라 하면 MAR → 14, MBR → 32

Check 17. 보조, 특수용도기억장치

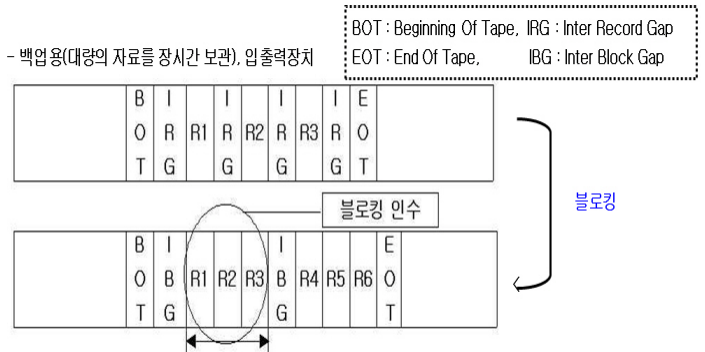
[출제빈도: 上]

1. 보조기억장치 ★★★★★

- 대규모의 기억용량을 갖는 장비로 구현
- data를 보관하였다가 주기억장치로 이동시키는 기능

2. 자기테이프 (순차접근) ★★★★★

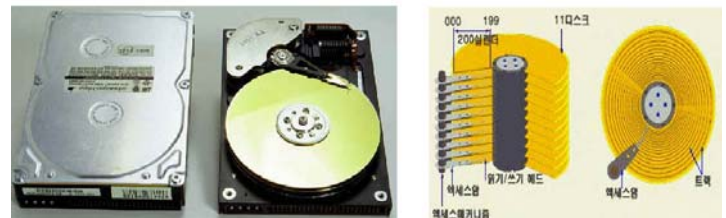
- 백업용(대량의 자료를 장시간 보관), 입출력장치



* 블로킹하는 이유 : 입출력 횟수(입출력 시간)를 줄이기 위해, 저장 공간을 절약할 수 있기 때문.

* 자기 테이프 Record 크기가 80자로서 블럭(Block)의 크기가 2,400자일 경우 블럭 팩터(Block Factor)는? 30

3. 자기 디스크 (직접 접근) ★★★★★



* 구성요소: 자기디스크, 액세스암, 헤드

* 실린더 수 = 트랙 수

4. 연관(연상) 기억장치 (Associative Memory) ★★★★★

- 자료를 찾을 때 주소(X), 기억된 내용의 일부를 이용.
- CAM (Contents addressable Memory)
- 가격이 비싸고, 속도가 빠르다.
- 기본요소 : 일치 지시기, 마스크 레지스터, 검색 데이터 레지스터

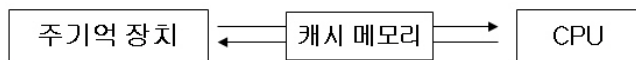
5. 복수 모듈 메모리 기법 (Memory Interleaving) ★★★★★

- 독자적으로 데이터를 저장할 수 있는 기억장치 모듈을 여러 개 가진 기억장치
- 기억장치에 접근하는 시간을 줄여 CPU와 속도차이를 줄이기 위한 기법
- 유료 Cycle동안 병렬 실행, 가격이 비싸고 속도가 빠르다.

- 각 모듈을 번갈아 가면서 접근
- 캐시 기억장치, 고속 DMA전송 등에서 많이 사용된다.

6. 캐시 메모리 (Cache Memory) ★★★★★

- CPU 속도와 메모리 속도 차이를 줄이기 위해 사용하는 고속 Buffer
- CPU와 주기억장치 사이에 위치
- 매핑방식
 - : 어소시어티브 매핑, 세트-어소시어티브 매핑, 직접 매핑



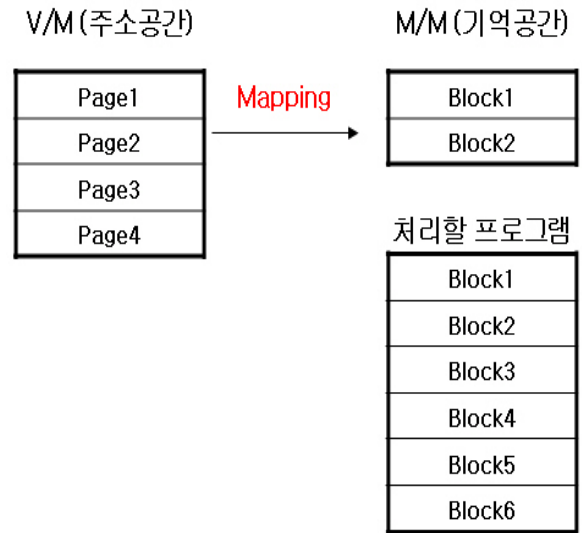
7. 가상기억장치(Virtual Memory) ★★★★★

- (직접)보조기억장치 일부를 마치 주기억장치인 것처럼 이용하는 운영체제의 메모리 운영 기법
- 주기억장치 용량 크게 (속도 X)

*관리기법

- 페이지징: 서로 같은 크기 부분
- 세그먼트: 서로 다른 크기의 부분

*Mapping: 가상기억장치에서 주기억장치로 자료의 페이지를 옮길 때 주소를 조정해 주는 것



Check 18. 병렬 컴퓨터

[출제빈도: 上]

1. 병렬 컴퓨터 ★★★★★

- 컴퓨터 성능 향상을 위해 여러 프로세서에서 동시에 여러 작업을 처리하는 것.

2. 병렬처리 컴퓨터 분류(폴린의 분류) ★★★★★

1) SISD (Single Instruction stream Single Data stream)

- 단일처리, 생산성 ↓, 병렬컴퓨터 X

2) SIMD (Single Instruction stream Multiple Data stream)

- 다중처리, Multiprogramming

3) MISD (Multiple Instruction stream Single Data stream)

- 이론적이며, 현재 사용하지 않는다.

4) MIMD (Multiple Instruction stream Multiple Data stream)