Microprocessor (W4)
- Central Processing Unit (CPU) -

Dong Min Kim
Department of IoT
Soonchunhyang University
dmk@sch.ac.kr

Contents

1 컴퓨터 기본 구조와 프로세서

- 컴퓨터의 3가지 핵심 장치 : 프로세서(Processor, CPU), 메모리, 입출력장치
- 버스(Bus) : 장치간에 주소, 데이터, 제어 신호를 전송하기 위한 연결 통로(연결선)

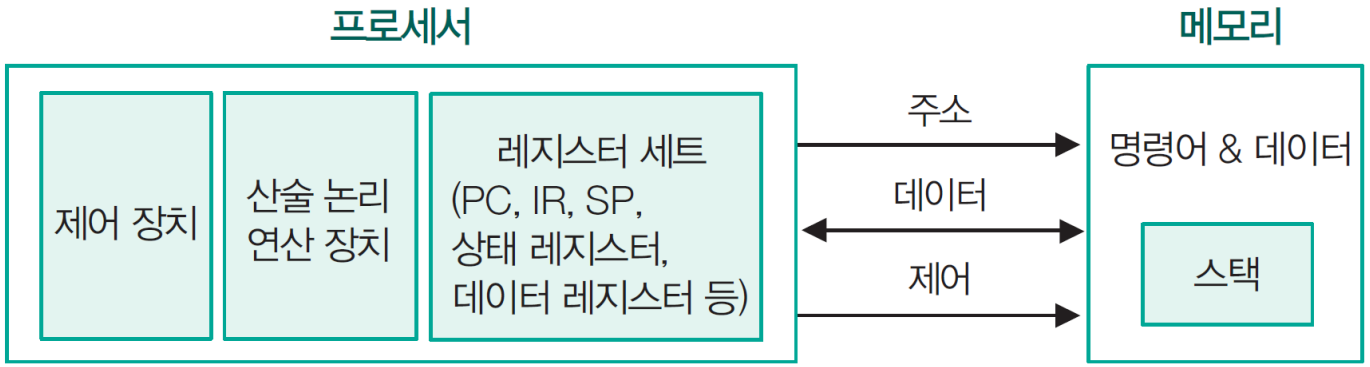


그림 4-1 폰 노이만 컴퓨터의 기본 구조

01 프로세서 구성과 동작

- 버스(Bus) : 장치간에 주소, 데이터, 제어 신호를 전송하기 위한 연결 통로(연결선)
  - 내부버스(internal bus) : 프로세서 내부의 장치 연결
  - 시스템 버스(system bus) : 핵심 장치 및 주변장치 연결

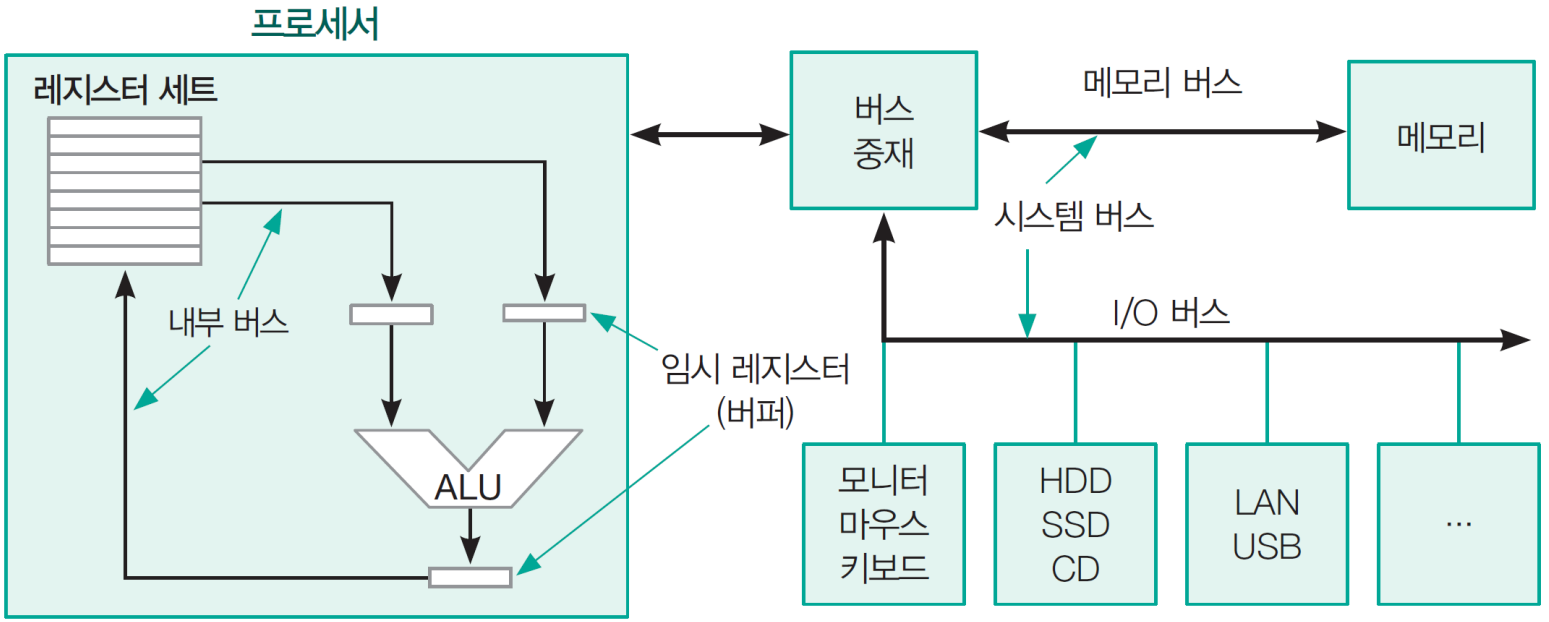


그림 4-2 버스 기반 컴퓨터 구조

2 프로세서 구성 요소

❖ 프로세서 3가지 구성 필수 구성요소

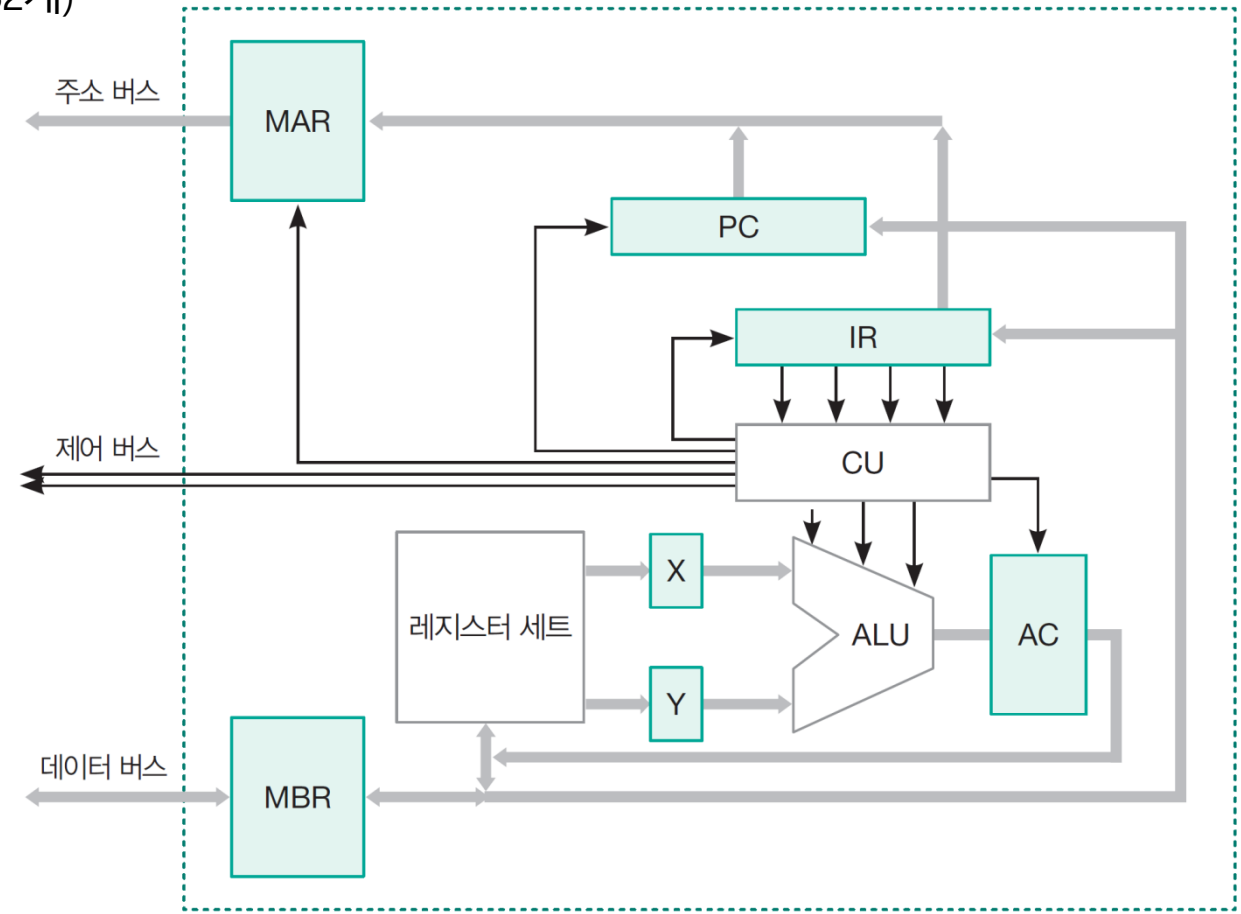
- 산술 논리 연산 장치(Arithmetic Logic Unit, ALU) : 산술 및 논리 연산 등 기본 연산을 수행
- 제어 장치 (Control Unit, CU) : 메모리에서 명령어를 가져와 해독하고 실행에 필요한 장치들을 제어하는 신호를 발생
- 레지스터 세트(register set) : 프로세서 내에 존재하는 용량은 작지만 매우 빠른 메모리, ALU의 연산과 관련된 데이터를 일시 저장하거나 특정 제어 정보 저장

  - 목적에 따라 특수 레지스터와 범용 레지스터로 분류

- 현재는 온칩 캐시(on-chip cache), 비디오 컨트롤러(video controller), 실수보조연산 프로세서(FPU) 등 다양한 장치 포함

3 프로세서 기본 구조

- 레지스터 세트(일반적으로 1~32개)
- ALU
- CU
- 이들 장치를 연결하는 버스로 구성



MAR Memory Address Register: 메모리 주소 레지스터
PC Program Counter: 프로그램 카운터
CU Control Unit: 제어 장치
AC ACcumulator: 누산기
MBR(MDR) Memory Buffer(Data) Register: 메모리 버퍼(데이터) 레지스터
IR Instruction Register: 명령 레지스터
ALU Arithmetic Logic Unit: 산술 논리 연산 장치

그림 4-3 프로세서 기본 구조

01 프로세서 구성과 동작

❖ ALU

- 덧셈, 뺄셈 등 연산을 수행하고, 그 결과를 누산기(Accumulator, AC)에 저장

❖ 프로세서 명령 분류

레지스터-메모리 명령	• 메모리 워드를 레지스터로 가져올(LOAD) 때 • 레지스터의 데이터를 메모리에 다시 저장(STORE)할 때
레지스터-레지스터 명령	• 레지스터에서 오퍼랜드 2개를 ALU의 입력 레지스터로 가져와 덧셈 또는 논리 AND 같은 몇 가지 연산을 수행하고 • 그 결과를 레지스터 중 하나에 다시 저장

4 프로세서 명령 실행

- 프로세서는 각 명령을 더 작은 마이크로 명령(microinstruction)들로 나누어 실행

  1단계 다음에 실행할 명령어를 메모리에서 읽어 명령 레지스터(IR)로 가져온다.

  2단계 프로그램 카운터(PC)는 그 다음 명령어의 주소로 변경된다.

  3단계 방금 가져온 명령어를 해독(decode)하고 유형을 결정한다.

  4단계 명령어가 메모리에 있는 데이터를 사용하는 경우 그 위치를 결정한다.

  5단계 필요한 경우 데이터를 레지스터로 가져온다.

  6단계 명령어를 실행한다.

  7단계 1단계로 이동하여 다음 명령어 실행을 시작한다.

- 이 단계를 요약하면 인출(fetch)-해독(decode)-실행(execute) 사이클로 구성 – 주 사이클(main cycle)

01 프로세서 구성과 동작

❖ **해독기**(microprogrammed control) : 하드웨어를 소프트웨어로 대체

- 고가의 고성능 컴퓨터는 하드웨어 추가 비용이 크게 부담되지 않아 저가 컴퓨터보다 많은 명령어를 갖게 됨
- 고가인 고성능 컴퓨터의 복잡한 명령어를 저가 컴퓨터에서 실행할 수 있게 하기 위함
- 모리스 윌크스(Maurice Wilkes)가 제안(1951년)
  - 1957년 SDSAC 1.5에 적용
- 1970년대 설계된 거의 모든 컴퓨터가 해독기를 기반
  - Cray-1 같은 매우 고가의 고성능 모델을 제외하고는 1970년대 후반에 해독기를 운영하는 프로세서가 보편적으로 보급
  - 복잡한 명령어에 대한 비용 절감, 훨씬 더 복잡한 명령어 연구
- 제어 기억 장치(control memory)라는 빠른 읽기 전용 메모리

02 산술 논리 연산 장치

❖ **산술 논리 연산 장치**(Arithmetic Logic Unit, ALU) : 산술 연산과 논리 연산
- 주로 정수 연산을 처리
- 부동 소수(Floating-point Number) 연산 : FPU(Floating-Point Unit)
- 최근에는 ALU가 부동 소수 연산까지 처리

❖ 산술 연산 : 덧셈, 뺄셈, 곱셈, 나눗셈, 증가, 감소, 보수

❖ 논리 연산 : AND, OR, NOT, XOR, 시프트(shift)

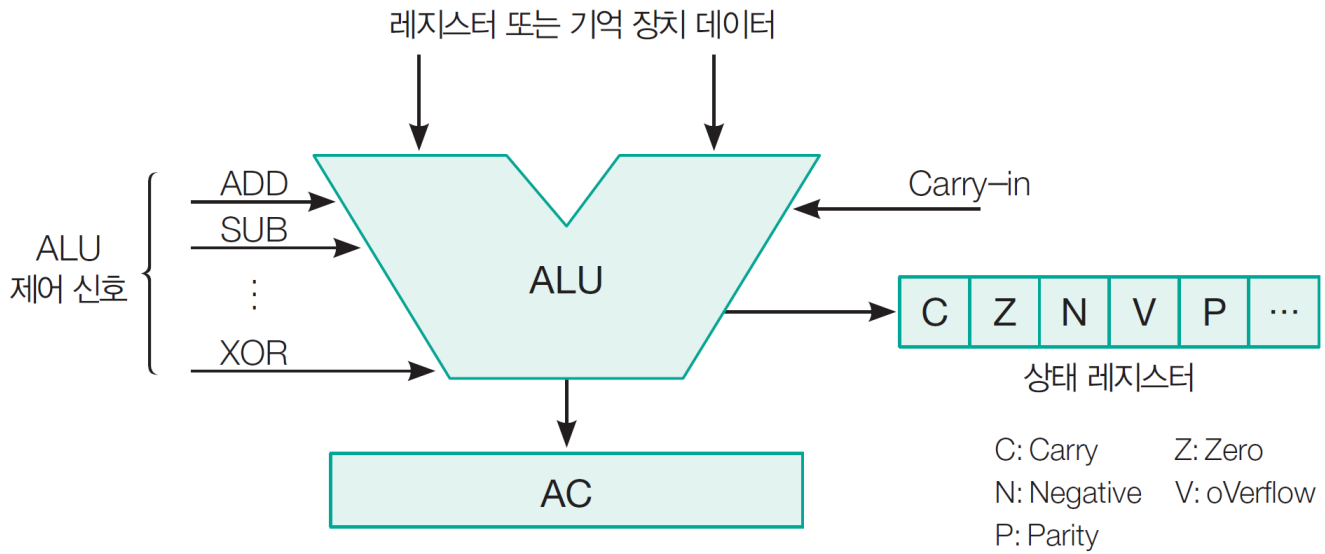


그림 4-4 ALU의 동작

1 산술 연산

표 4-1 산술 연산

연산	8비트 연산	
	동작	설명
ADD	X ← A + B	A와 B를 더한다.
SUB	X ← A + (~B + 1)	A + (B의 2의 보수)
MUL	X ← A * B	A와 B를 곱한다.
DIV	X ← A / B	A와 B를 나눈다.
INC	X ← A + 1	A를 1 증가시킨다.
DEC	X ← A − 1(0xFF)	A를 1 감소시킨다.
NEG	X ← ~A + 1	A의 2의 보수다.

❖ Booth Algorithm

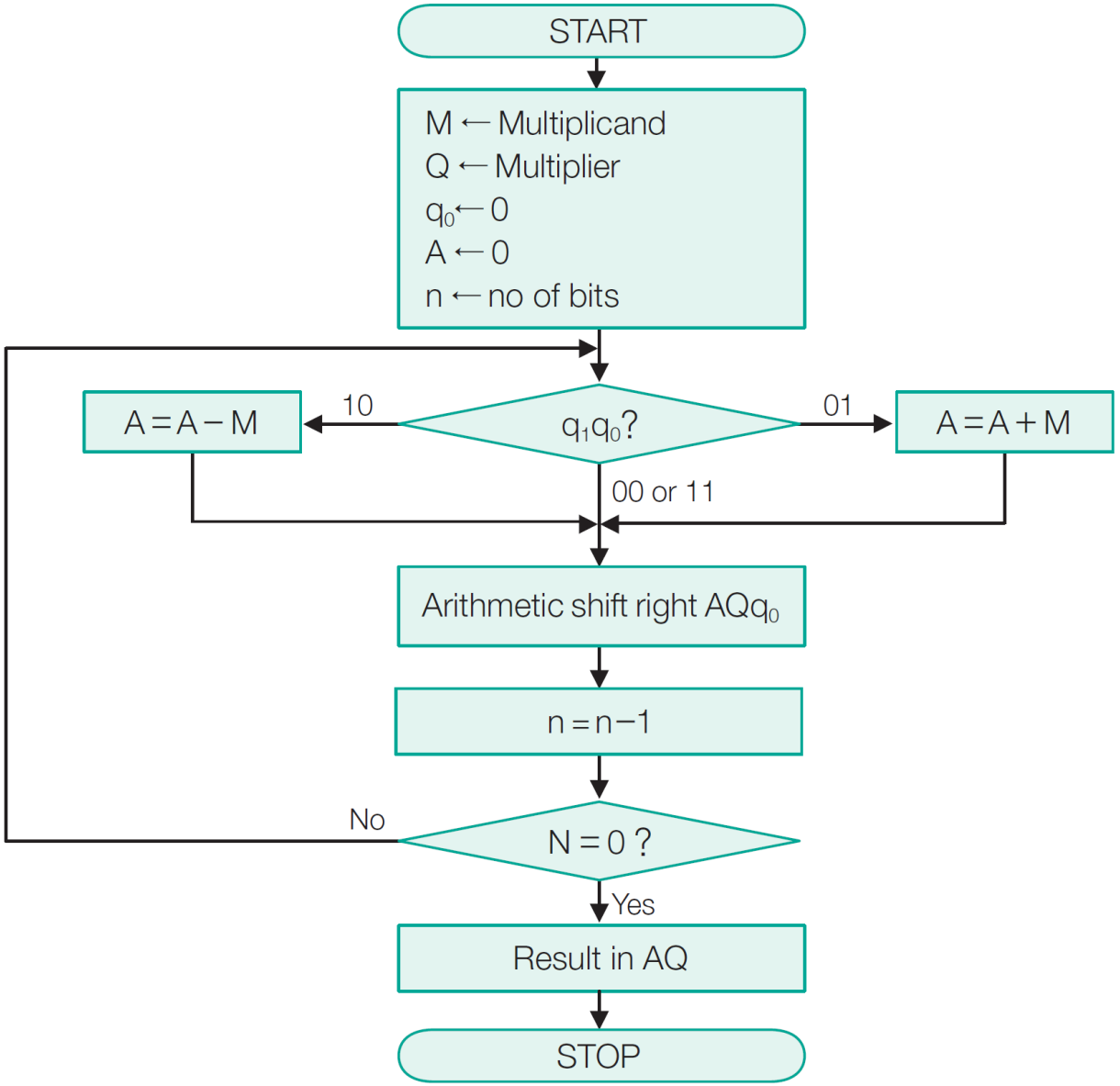


그림 4-5 부스 알고리즘 순서도

❖ **Booth Algorithm 예** : (-7) * (+3)

n	A	$Q(q_4q_3q_2q_1)$	q_0	설명
4	0000	0011	0	초기 상태
	0111	0011	0	q_1q_0이 10이므로 A=A−M=0000+0111
3	0011	1001	1	AQq_0을 오른쪽 산술 시프트
2	0001	1100	1	q_1q_0이 11이므로 연산 없이 AQq_0을 오른쪽 산술 시프트
	1010	1100	1	q_1q_0이 01이므로 A=A+M=0001+1001
1	1101	0110	0	AQq_0를 ASR
0	1110	1011	0	q_1q_0가 00이므로 연산 없이 AQq_0을 오른쪽 산술 시프트

❖ **Booth Algorithm 예** : 5 * (-4)

n	A	$Q(q_4q_3q_2q_1)$	q_0	설명
4	0000	1100	0	초기 상태
3	0000	0110	0	q_1q_0이 00이므로 연산 없이 AQq_0을 오른쪽 산술 시프트
2	0000	0011	0	q_1q_0이 00이므로 연산 없이 AQq_0을 A오른쪽 산술 시프트
1	1011	0011	0	q_1q_0이 10이므로 A=A–M=0000+1011
	1101	1001	1	AQq_0을 오른쪽 산술 시프트
0	1110	1100	1	q_1q_0이 11이므로 연산 없이 AQq_0을 오른쪽 산술 시프트

2 논리 연산과 산술 시프트 연산

표 4-2 논리 연산

연산	8비트 연산	
	동작	설명
AND	X←A&B	A와 B를 비트 단위로 AND 연산한다.
OR	X←A|B	A와 B를 비트 단위로 OR 연산한다.
NOT	X←~A	A의 1의 보수를 만든다.
XOR	X←A^B	A와 B를 비트 단위로 XOR 연산한다.
ASL	X←A≪n	왼쪽으로 n비트 시프트(LSL과 같다.)
ASR	X←A≫n, A[7]←A[7]	오른쪽으로 n비트 시프트(부호 비트는 그대로 유지한다.)
LSL	X←A≪n	왼쪽으로 n비트 시프트
LSR	X←A≫n	오른쪽으로 n비트 시프트
ROL	X←A≪1, A[0]←A[7]	왼쪽으로 1비트 회전 시프트, MSB는 LSB로 시프트
ROR	X←A≫1, A[7]←A[0]	오른쪽으로 1비트 회전 시프트, LSB는 MSB로 시프트
ROLC	X←A≪1, C←A[7], A[0]←C	캐리도 함께 왼쪽으로 1비트 회전 시프트
RORC	X←A≫1, C←A[0], A[7]←C	캐리도 함께 오른쪽으로 1비트 회전 시프트

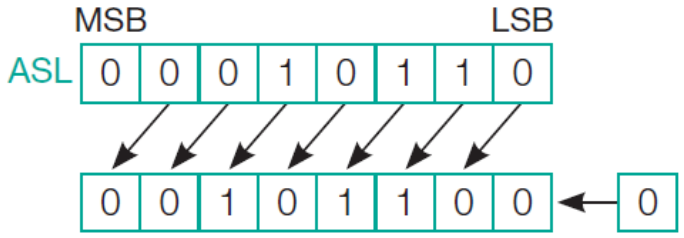
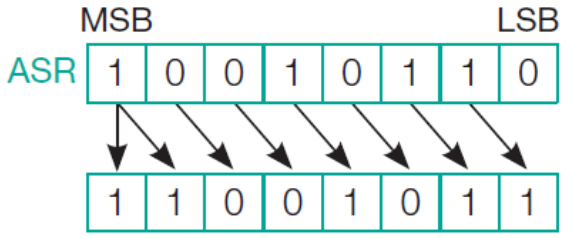
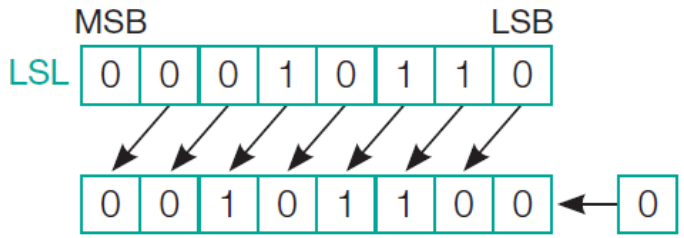
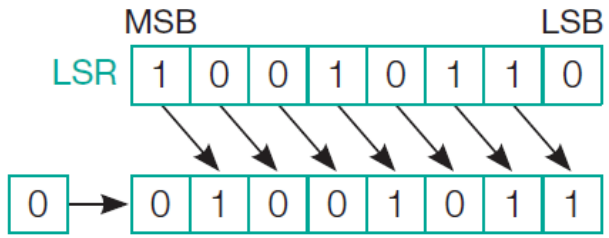
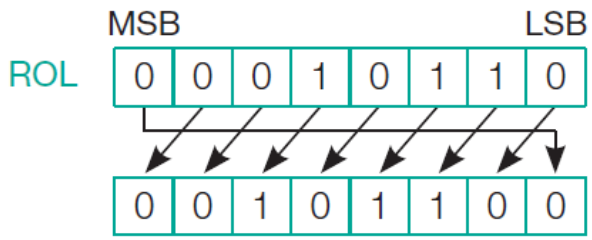
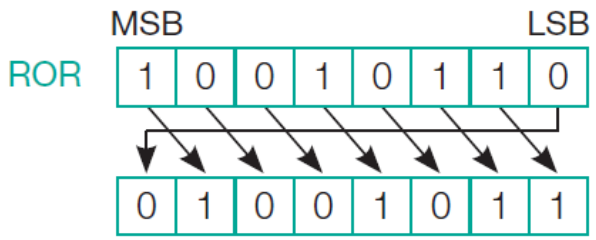
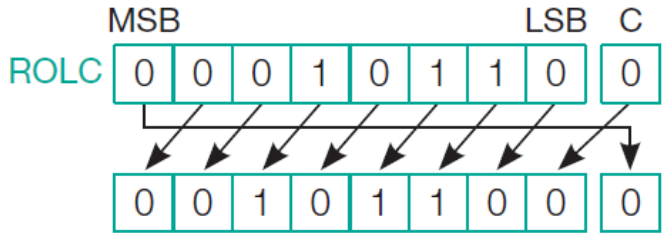
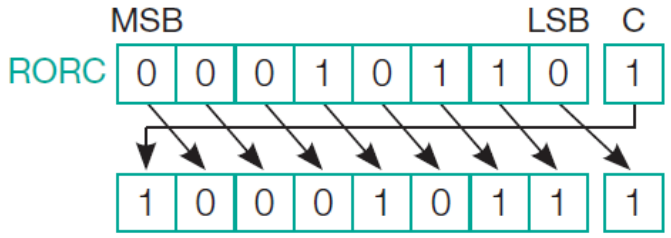
❖ **논리 연산 예 1** : A=46=$00101110_{(2)}$, B=-75=$10110101_{(2)}$

A AND B			A OR B			A XOR B		
	00101110	46		00101110	46		00101110	46
&	10110101	−75	|	10110101	−75	^	11111111	−128
	00100100	36		10111111	−65		11010001	−47

❖ **논리 연산 예 2**

A AND B		A OR B	
00101110		00001110	
& 00001111	상위 4비트 삭제	| 10110000	상위 4비트 값 설정
00001110		10111110	

❖ 시프트 연산 예

연산	왼쪽	오른쪽
산술 시프트		
논리 시프트		
회전 시프트		
캐리와 함께 회전 시프트		

Summary

- 컴퓨터 프로세서의 기본 구조와 명령 실행 과정 이해
- ALU 구조를 이해하고 프로세서에서의 산술 및 논리 연산을 학습