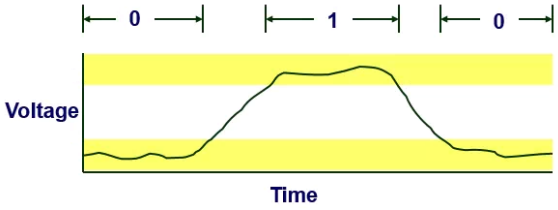
4장. 프로세서 구조

4.1 Y86-64 명령어 집합
4.2 논리 설계와 하드웨어 제어 언어 HCL

4.3 순차적 Y86-64 구현
4.4 파이프라이닝의 일반 원리
3.5 파이프라인형 Y86-64의 구현

4.2 논리 설계와 하드웨어 제어 언어 HCL

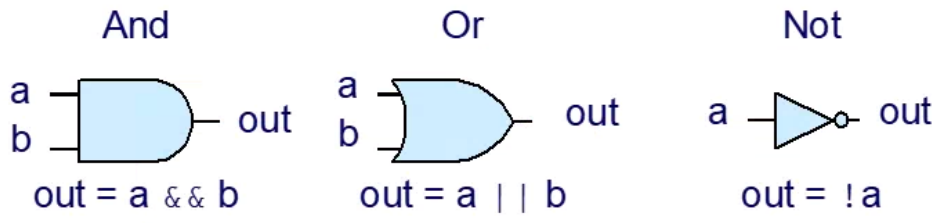
논리 설계 (Logic Design)와 HCL 개요

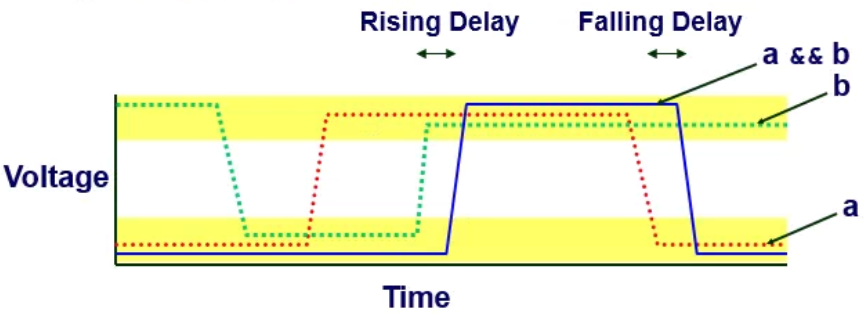


- 디지털 신호
  - 전기적 신호의 전압 수준에 따라 1과 0을 표현
- 디지털 시스템 컴포넌트
  - 조합 논리회로 (combinational logic circuit)
    - 비트 연산 (부울 함수)을 수행
  - 메모리 소자
    - 상태를 저장
    - 조합 논리회로와 함께 순차 논리 회로 (sequential logic circuit) 구성
  - 클럭 신호
    - 메모리 소자의 갱신을 동기화
- HCL (Hardware Control Language)
  - 프로세서 설계 시 제어 회로를 기술하는 하드웨어 제어 언어
  - 부울 연산을 C 언어의 논리 연산과 유사하게 표현

논리 게이트

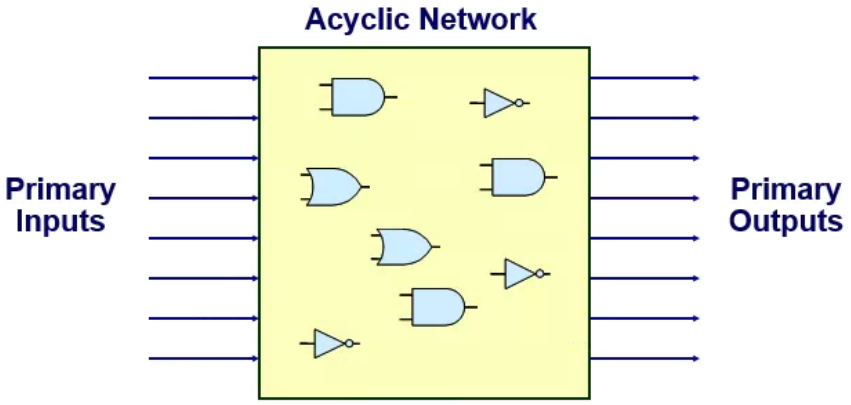
- 입력에 대한 부울 함수 적용 결과를 출력
  - 입력 값에 변화에 따라 일정 시간 지연 후 응답





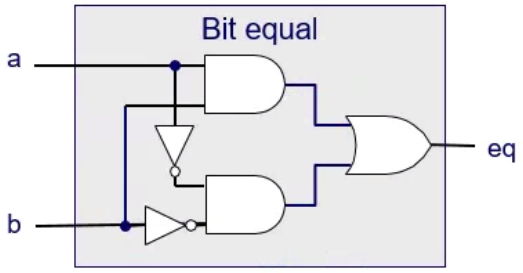
조합 논리회로

- 여러 개의 논리 게이트들을 조합하여 구성
  - 논리 게이트들 간 네트워크이 비순환(acyclic) 회로



비트 수준 동일성 논리회로

- 두 입력 비트가 같으면 1을 출력

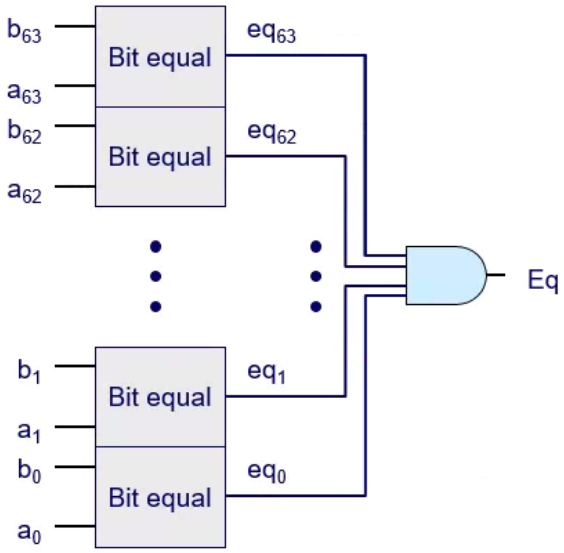


HCL Expression

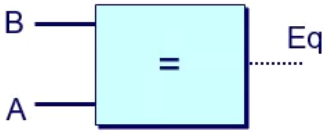
```
bool eq = (a&&b) || (!a&&!b)
```

워드 수준 동일성 논리회로

- 64비트 워드의 동일성 검사



Word-Level Representation

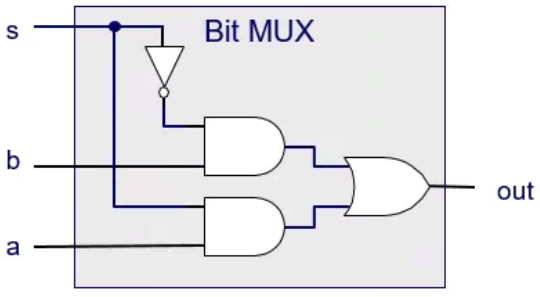


HCL Representation

```
bool Eq = (A == B)
```


비트 수준 멀티플렉서 논리회로

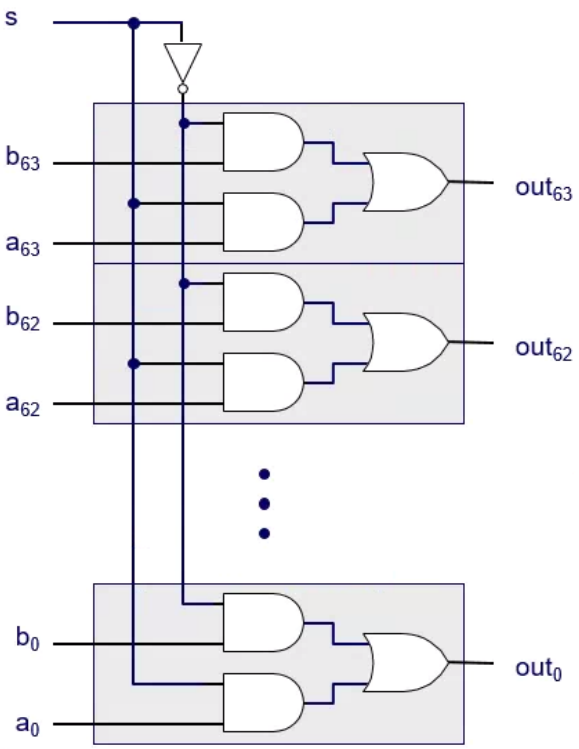
- 제어 신호 s가 1이면 a, 0이면 b 출력
  - 2 x 1 MUX (Multiplexer)



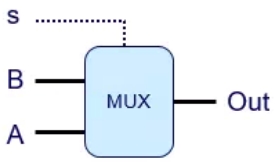
HCL Expression

```
bool out = (s&&a)||(!s&&b)
```

워드 수준 멀티플렉서 논리회로



Word-Level Representation



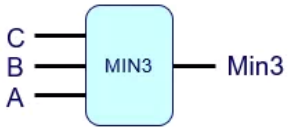
HCL Representation

```
int Out = [
  s : A;
  1 : B;
];
```

- HCL의 케이스 수식 (case expression)
  - 첫번째로 1이 되는 케이스를 선택
  - C의 switch와 달리 각 케이스가 상호배타적이 아님

HCL 워드 수준 코드 예

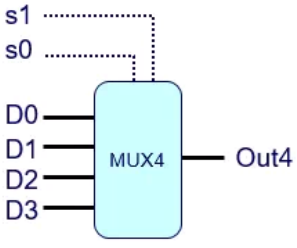
Minimum of 3 Words




```
int Min3 = [
  A < B && A < C : A;
  B < A && B < C : B;
  1              : C;
];
```

- 세 개의 입력 중 최소값을 출력
- HCL 케이스 표현
  - 최소한 마지막 케이스는 매치

4-Way Multiplexor

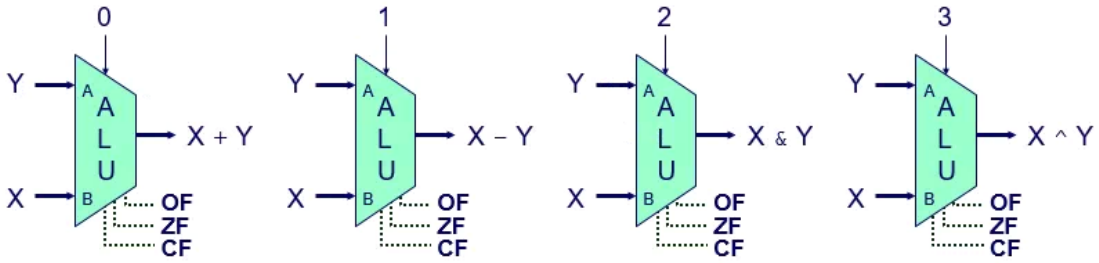



```
int Out4 = [
  !s1&&!s0: D0;
  !s1     : D1;
  !s0     : D2;
  1       : D3;
];
```

- 2비트 제어신호로 4개의 입력 중 하나를 선택
- HCL 케이스 표현
  - 순차적인 매치를 가정하여 테스트를 단순화

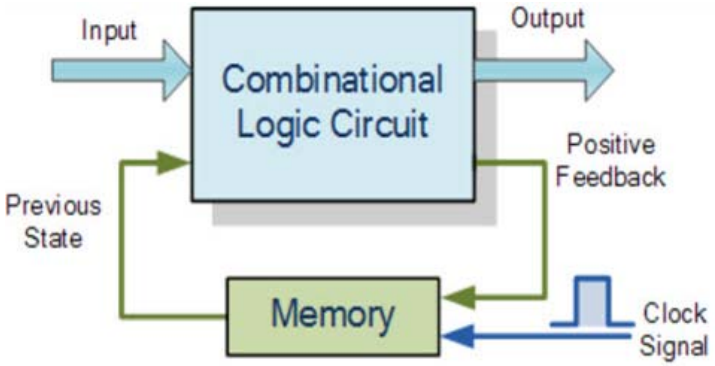
산술/논리 유닛

- 산술/논리 유닛 (Arithmetic Logic Unit, ALU)
  - 산술 및 논리 연산을 수행하는 조합 논리회로
  - 제어신호가 계산되는 연산을 선택
    - Y86-64에서는 4개의 산술/논리 연산
  - 조건 코드도 세팅



순차 논리회로

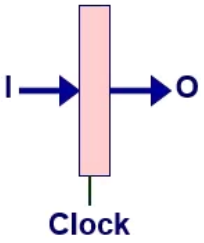
- 조합 논리회로는 정보를 저장하지 않음
- 순차 논리회로 (sequential logic circuit)
  - 저장되는 상태를 가짐
    - 상태 머신 (state machine)
  - 메모리 소자에 상태를저장하며 클럭에 의해 새로운 값이 저장



레지스터

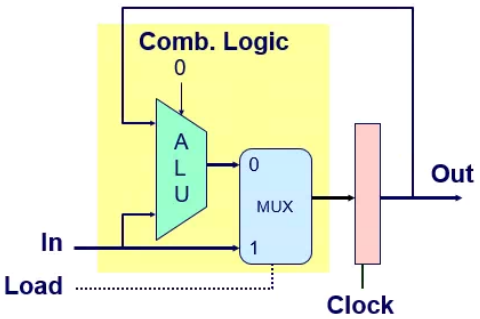
- 프로세서 설계 시 두가지 종류의 메모리 장치 소개
  - 레지스터 (Register)
  - 랜덤 액세스 메모리 (Random Access memory)
    - 단순히 메모리로 지칭

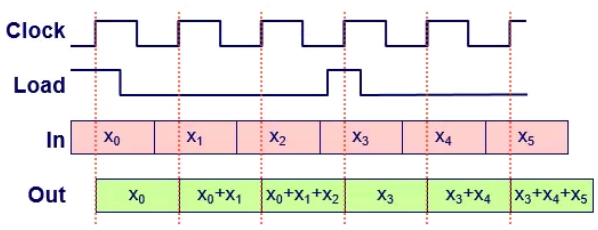
- 레지스터 (Register)
  - 어셈블리 코드에서 소개된 레지스터와 약간 다름
    - 하드웨어 레지스터와 프로그램 레지스터로 구분



상태 머신 예

- 누산기 (accumulator) 예
  - 각 클럭 사이클마다 새로운 값이나 누적합을 저장

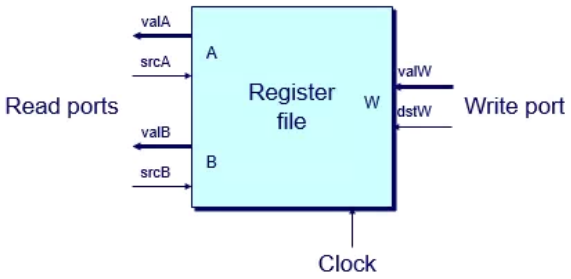




레지스터 파일 (Register File)

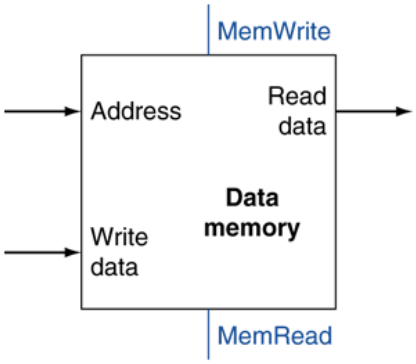
- 메모리에 여러 개의 워드를 저장
  - 읽기 또는 쓰기를 할 워드를 선택하는 주소 입력을 기술

- 레지스터 파일
  - 프로그램 레지스터의 값들을 저장
    - %rax, %rsp, ...
  - 레지스 터 구분자가 주소 역할
    - ID 15 (0xF)는 어떤 읽기 또는 쓰기도 수행하지 않음
  - 여러개의 포트
    - 한 (클럭) 사이클에 여러 개의 워드의 읽기, 쓰기가 가능
      - 각 포트는 분리된 주소와 데이터 입력/출력을 가짐



랜덤 액세스 메모리

- 프로세서에서 프로그램 데이터를 저장
- 한개의 주소 입력, 쓰기 데이터 제어신호 , 읽기 데이터 출력



하드웨어 제어 언어 (Hardware Control Language, HCL)

- 아주 간단한 하드웨어 기술 언어
- 제한된 하드웨어 동작 기술

Data Types

- `bool`: Boolean
  - a, b, c, ...
- `int`: words
  - A, B, C, ...
  - Does not specify word size---bytes, 64-bit words, ...

Statements

- `bool a = bool-expr ;`
- `int A = int-expr ;`

HCL 연산

Boolean Expressions

- Logic Operations
  - `a && b, a || b, !a`
- Word Comparisons
  - `A == B, A != B, A < B, A <= B, A >= B, A > B`
- Set Membership
  - `A in { B, C, D }`
    - » Same as `A == B || A == C || A == D`

Word Expressions

- Case expressions
  - `[ a : A; b : B; c : C ]`
  - Evaluate test expressions a, b, c, ... in sequence
  - Return word expression A, B, C, ... for first successful test

요 약

요 약

- 조합 논리회로
  - 부울 함수를 계산
  - 입력에 따라 출력 응답
- 메모리 장치
  - 상태를 저장하며 순차 논리회로를 구성
    - 클럭이 상승할 때 새로운 값이 저장
  - 레지스터
    - 단일 워드를 저장
    - 클럭이 상승할 때 저장
  - 레지스터 파일
    - 여러 개의 워드를 저장
    - 여러 개의 읽기 또는 쓰기 포트
  - 랜덤 액세스 메모리
    - 한개의 주소 입력, 쓰기 데이터 제어신호 , 읽기 데이터 출력