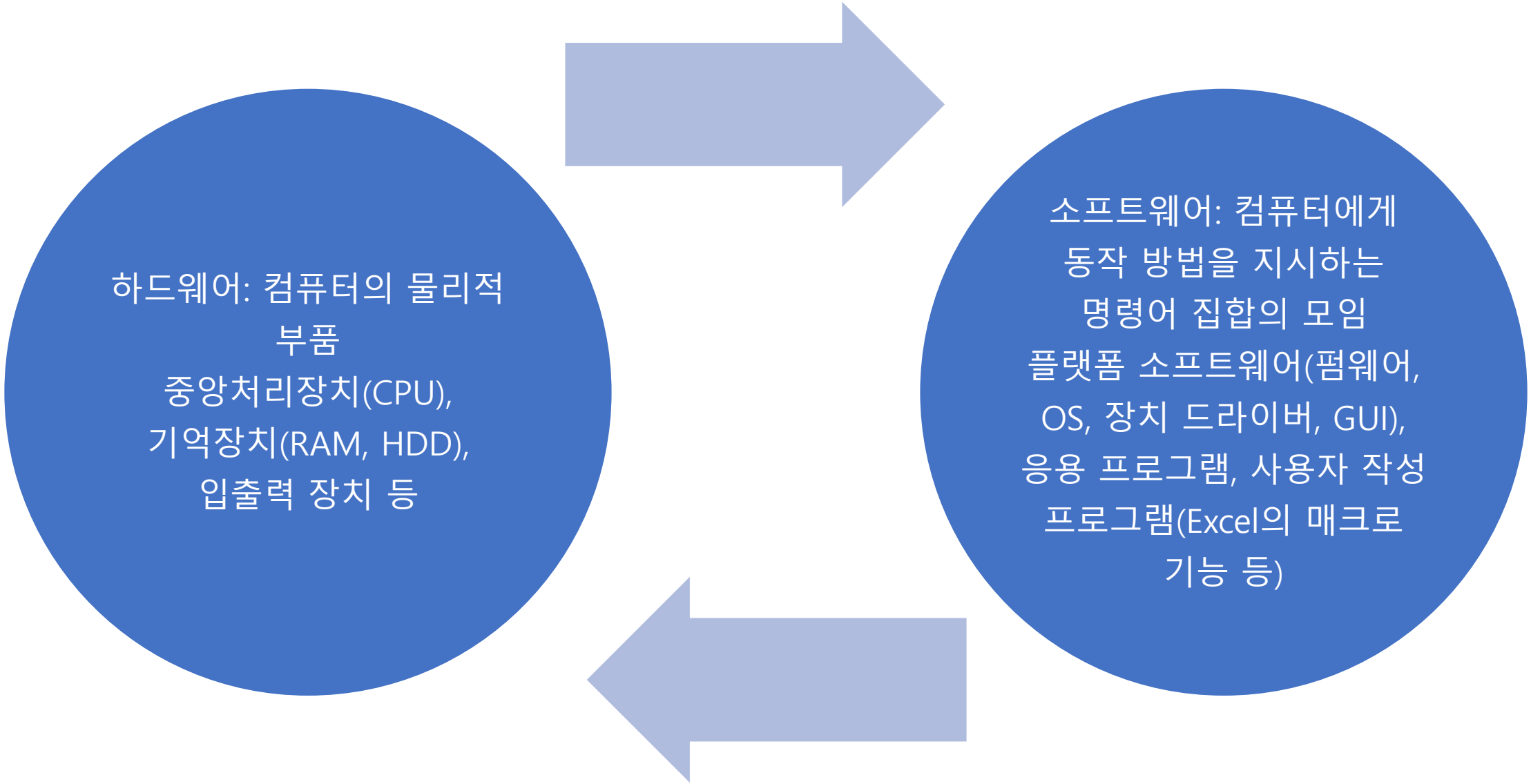
남재현

컴퓨터 구조

컴퓨터의 구성


하드웨어: 컴퓨터의 물리적 부품
중앙처리장치(CPU), 기억장치(RAM, HDD), 입출력 장치 등
소프트웨어: 컴퓨터에게 동작 방법을 지시하는 명령어 집합의 모임
플랫폼 소프트웨어(펌웨어, OS, 장치 드라이버, GUI), 응용 프로그램, 사용자 작성 프로그램(Excel의 매크로 기능 등)

CPU의 구성요소

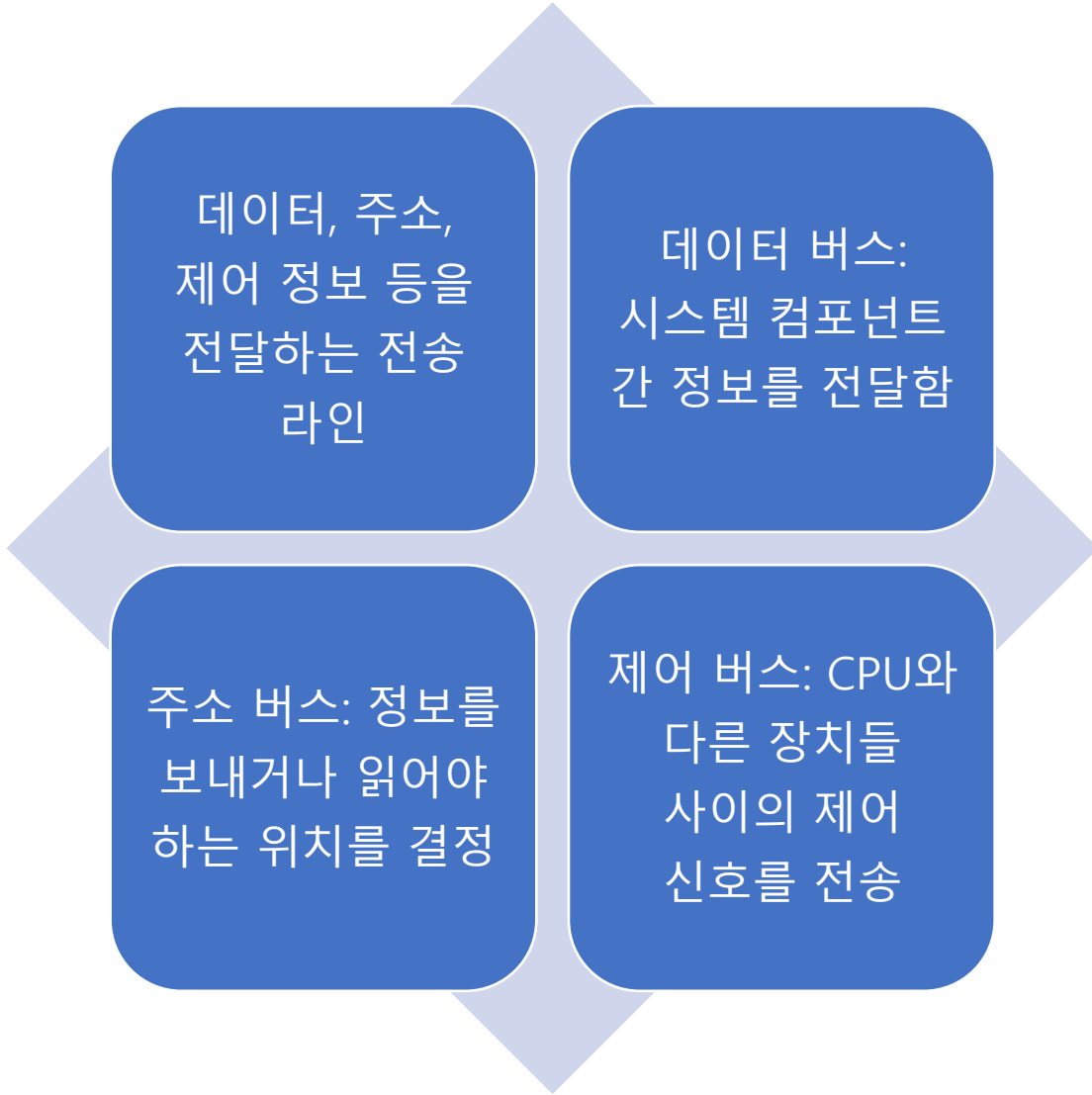
ALU(연산장치): 각종 산술 연산들과 논리연산들을 수행

Register(레지스터): CPU 내부의 데이터를 일시적으로 기억해 두는 고속의 영역

Control Unit(제어장치): 프로그램 코드(명령어)를 해석하고 그것을 실행하기 위한 제어 신호들을 발생시킴

내부 CPU 버스: ALU와 레지스터 사이의 데이터 이동 경로

시스템 버스 (System Bus)



CPU 성능

응답시간(response time) : 실행시간(execution time)이라고도 한다. 컴퓨터가 테스크를 완료하기까지의 총 소요시간으로 디스크 접근, 메모리 접근, 입출력 작업, 운영체제 오버헤드 및 CPU 시간을 다 포함한다.

처리량(throughput): 대역폭(bandwidth)라고도 한다. 단위시간당 완료하는 태스크의 수를 나타낸다.

성능 = 1 / 실행시간

프로그램의 CPU 실행시간 = 프로그램의 CPU 클럭 사이클 수 / 클럭 속도

CPU 클럭 사이클 수 = 명령어 수 × 명령어 당 평균 클럭 사이클 수

명령어 당 평균 클럭 사이클 수를 CPI라고 한다.

CPU 시간 = 명령어 개수 × CPI × 클럭 사이클 시간

CPU의 명령어 주기 (Instruction Cycle)

이전의 명령어의 실행이 끝난 뒤부터 다음 명령어의 실행이 끝나기까지 걸리는 시간

인출(Fetch): 메모리에서 데이터를 로드하여 CPU의 레지스터에 적재하는 과정

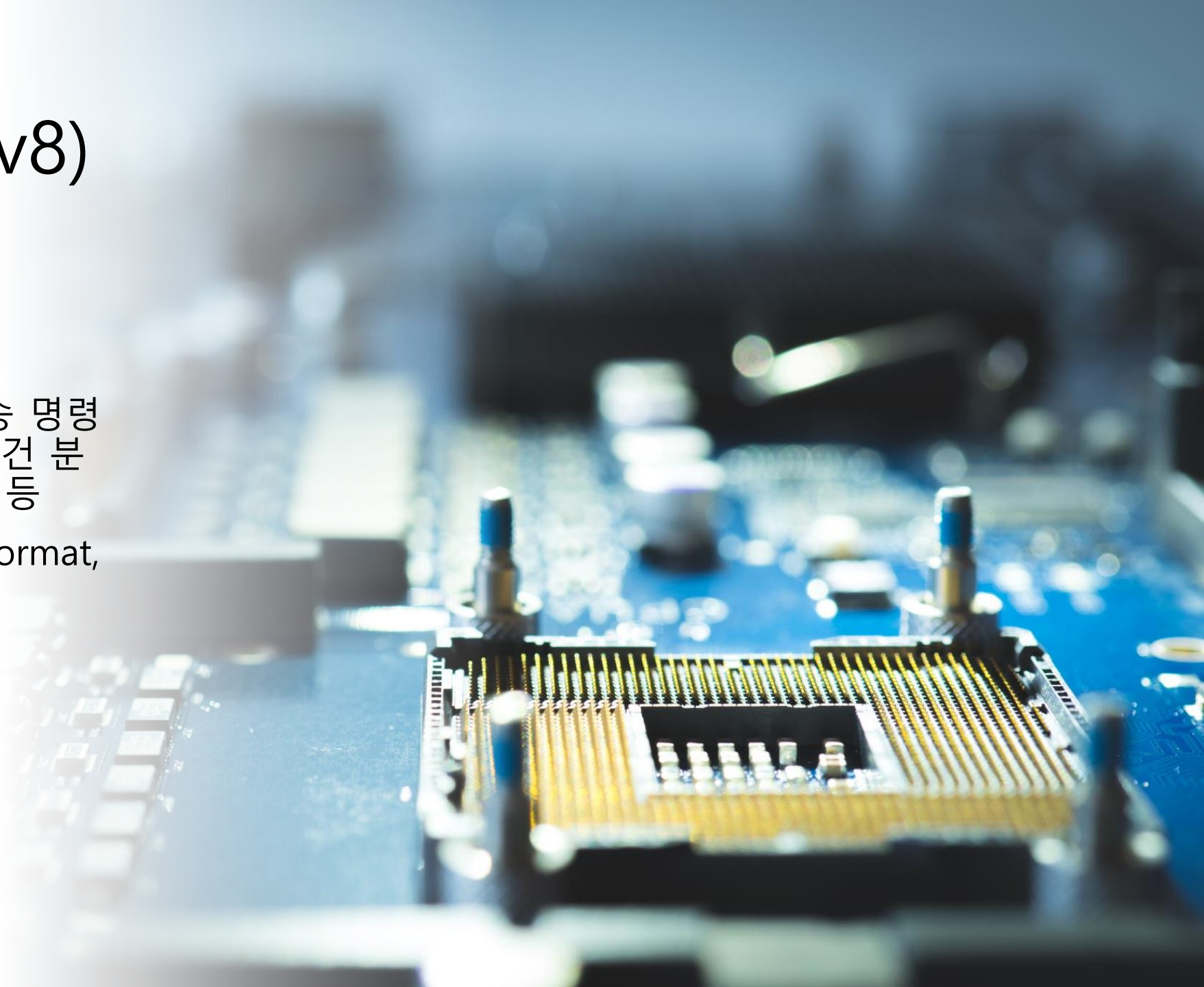
간접(Indirect): CPU가 참조한 메모리의 값이 주소 값이면 그 주소 값으로 다시 메모리를 참조함

실행(Execution): 명령과 데이터로 CPU가 연산을 수행

인터럽트(Interupt): 이벤트나 문제 등이 발생했을 때 업무 처리가 계속될 수 있도록 하는 기능

LEGv8 (ARMv8) 명령어 집합

- 종류:
  산술 명령어, 데이터전송 명령어, 논리연산 명령어, 조건 분기 명령어, 분기 명령어 등
- R-format, I-format, D-format, B-format 등



LEGv8 산술 명령어

- 형식을 반드시 지켜야 함
  ADD a, b, c → b와 c의 합을 a에 넣는다

  변수 b, c, d, e의 합을 a에 넣기
  → ADD a, b, c
  ADD a, a, d
  ADD a, a, e
- 설계 원칙 1: 간단하기 위해서는 규칙적인 것이 좋다.
  (피연산자의 수가 가변적이면 하드웨어가 복잡해진다)

피연산자

- 산술 명령어의 피연산자에는 제약이 있음-> 레지스터에 있는 것만을 이용 가능함
- 레지스터의 크기는 64비트(더블워드)임
- 산술 명령어의 모든 연산자는 32개의 64비트 레지스터 중 하나이어야 함 (LEGv8의 관례 상 X0~X31로 불림)
- 설계 원칙 2: 작은 것이 더 빠르다. (레지스터가 아주 많아지면 전기 신호가 더 멀리까지 전달되어야 하므로 클럭 사이클 시간이 길어진다)

피연산자

- 메모리 데이터 적재 또는 저장 명령어도 필요함
- 상수를 사용하는 명령어도 필요함
  → ADDI X22, X22, #4
  (X22에 4를 더한 값을 X22에 넣는다)
- 레지스터 XZR의 값은 0으로 고정
  (자주 생기는 것을 빠르게)

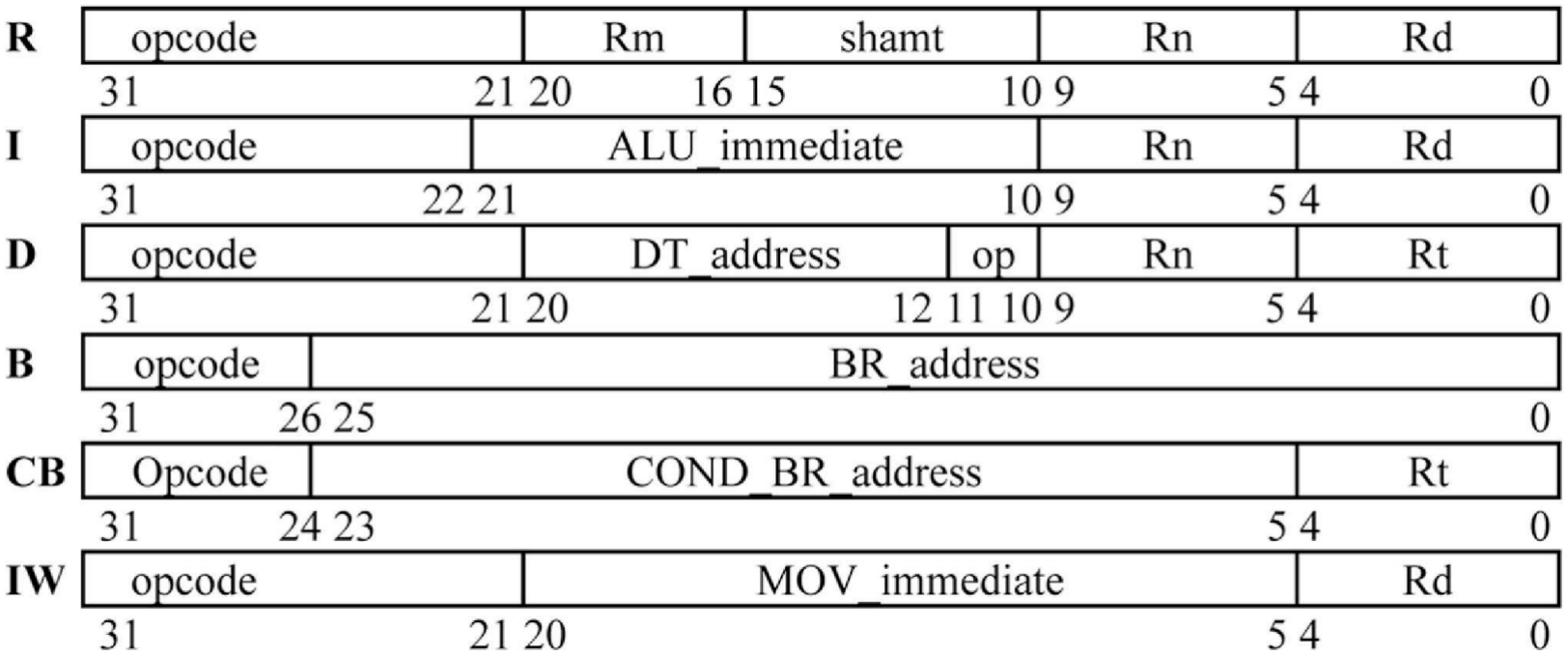
명령어의 컴퓨터 내부 표현

opcode	Rm	shamt	Rn	Rd
11 bits	5 bits	6 bits	5 bits	5 bits

- ADD X9, X20, X21
  → 10001011000 10101 000000 10100 01001
- Opcode: 명령어가 실행할 연산의 종류
- Rm: 두 번째 근원지(source) 피연산자 레지스터
- Shamt: 자리이동(shift) 양
- Rn: 첫 번째 근원지 피연산자 레지스터
- Rd: 목적지(destination) 레지스터(연산 결과 저장)
- 설계원칙 1(규칙적인 것이 좋음)에 따라 위 필드의 조합을 지켜야 하지만, 이것보다 필드 길이가 더 길어야 하는 경우에는 문제가 생길 수 있음 (예컨데 load register 명령어는 레지스터 필드 두 개와 상수 필드 하나가 필요하지만 상수 필드의 길이가 32 bits 보다 짧아서 한계가 있음)

  설계 원칙 3: 좋은 설계에는 적당한 절충이 필요함
  하지만 LEGv8 설계자들은 모든 명령어의 길이를 같게 하되, 명령어 종류에 따라 형식을 다르게 함 -> R형식 뿐만 아니라 D형식, I형식도 만듦

LEGv8 명령어 형식


R
opcode
Rm
shamt
Rn
Rd
31
21 20
16 15
10 9
5 4
0
I
opcode
ALU_immediate
Rn
Rd
31
22 21
10 9
5 4
0
D
opcode
DT_address
op
Rn
Rt
31
21 20
12 11 10 9
5 4
0
B
opcode
BR_address
31
26 25
0
CB
Opcode
COND_BR_address
Rt
31
24 23
5 4
0
IW
opcode
MOV_immediate
Rd
31
21 20
5 4
0

하드웨어의 프로시저 지원

- 프로시저(procedure): 제공되는 인수에 따라서 특정 작업을 수행하는 서브루틴
- LEGv8 소프트웨어는 프로시저를 호출할 때 다음의 레지스터 할당 관례를 따름
  X0-X7: 전달할 인수와 결과값을 가지고 있는 인수 레지스터 8개
  LR (X30): 호출한 곳으로 되돌아가기 위한 복귀 주소를 가지고 있는 레지스터 1개
- BR LR
  위 명령어는 레지스터 LR에 저장되어 있는 주소로 분기하라는 명령어
- 프로그램 카운터(Program Counter): 현재 수행되는 명령어의 주소를 기억하는 레지스터; 약어로 PC라고 부름

더 많은 프로시저의 사용

스택 포인터(Stack Pointer): 가장 최근에 스택에 할당된 주소를 가리키는 값; 보통 약어로 SP라고 함

레지스터에 들어가지 못할 만큼 큰 배열이나 구조체 같은 지역 변수를 저장하는데도 스택이 사용됨