주제/단락	내용
명령어 세트 정의	특정 CPU를 위해 정의된 명령어들의 집합이다.
명령어 세트 설계 요소	연산 종류, 데이터 형태(길이와 표현), 명령어 형식, 주소지정 방식을 결정해야 한다.
연산 종류 범주	데이터 전송(MOVE, LOAD, STORE), 산술 연산, 논리 연산, 입출력, 프로그램 제어(분기, 서브루틴 호출)로 구성된다.
연산 코드	수행될 연산을 지정하며 예로 LOAD, ADD 등이 있다.
오퍼랜드 개념	연산에 필요한 데이터 또는 그 주소로, 입력 오퍼랜드 1~2개와 결과 오퍼랜드 1개를 포함하며 위치는 레지스터, 메모리, I/O가 될 수 있다.
다음 명령어 주소	현재 명령어 종료 후 인출할 위치를 지정하며, 분기나 호출 등 순서 변경 시 필요하다.
명령어 형식 정의	명령어 내 필드 수와 배치, 각 필드의 비트 수를 정의한다.
필드의 의미	명령어 구성 요소에 할당된 비트들의 그룹을 말한다.
명령어 길이와 단어	일반적으로 명령어 길이는 단어 길이와 같다.
연산 코드 필드 길이	연산 개수를 결정하며 4비트면 16가지, 5비트면 32가지 연산을 정의할 수 있으나 다른 필드가 줄어든다.
오퍼랜드 필드 길이	데이터 범위, 메모리 주소 지정 가능 용량, 레지스터 개수 범위를 좌우한다.
혼합 오퍼랜드 예	오퍼랜드1이 레지스터 번호(4비트)면 16개 레지스터, 오퍼랜드2가 8비트 주소면 0~255번지를 지정한다.
통합 오퍼랜드 범위	데이터가 12비트 2의 보수면 -2048~+2047, 주소라면 2^12=4096개 주소를 직접 지정한다.
0주소 명령어	스택 등으로 오퍼랜드와 결과 위치가 묵시적으로 정해지며 후위 표기법을 사용한다.
1주소 명령어	오퍼랜드 한 개를 포함하고 다른 한 오퍼랜드와 결과는 누산기 AC로 묵시 지정된다.
2주소 명령어	두 오퍼랜드를 포함하며 레지스터-레지스터, 레지스터-메모리 조합을 지원한다.
3주소 명령어	세 오퍼랜드를 포함해 입력 데이터를 보존하면서 결과를 별도에 저장한다.

주제/단락	내용
1주소 16비트 예제	연산코드 5비트면 주소필드 11비트로 주소지정 가능한 용량은 2^11×1B=2048B이다.
2주소 16비트 예제	연산코드 5비트, 레지스터 8개인 경우 레지스터-레지스터, 레지스터-메모리 형식을 설계한다.
3주소 16비트 예제	필요한 연산 12개, 레지스터 10개 조건에서 ADD R1,R2,R3 형식을 정의한다.
수식 계산 명령 집합	ADD, SUB, MUL, DIV, MOV, LOAD, STOR 니모닉을 사용한다.
1주소 수식 프로그램	LOAD A; ADD B; STOR T; LOAD C; SUB D; MUL T; STOR X로 구성되고 길이는 7이다.
2주소 수식 프로그램	MOV R1,A; ADD R1,B; MOV R2,C; SUB R2,D; MUL R1,R2; MOV X,R1로 길이는 6이다.
3주소 수식 프로그램	ADD R1,A,B; SUB R2,C,D; MUL X,R1,R2로 길이는 3이며 입력 데이터가 보존되나 해독이 복잡하다.
주소지정 방식 정의	명령어 실행에 필요한 오퍼랜드의 유효 주소를 결정하는 방법으로 CPU마다 종류와 수가 다르다.
다양한 방식의 이유	명령어 비트 수가 제한되어 있어 다양한 주소 결정과 큰 메모리 사용을 가능하게 하려는 목적이다.
오퍼랜드 필드 내용	메모리 주소, 레지스터 번호, 즉시 데이터가 될 수 있다.
기호 정의	EA는 유효 주소, A는 주소필드 내용, R은 레지스터 번호, (A)는 메모리 A번지 내용, (R)은 레지스터 R 내용이다.
주소지정 방식 목록	직접, 간접, 묵시적, 즉시, 레지스터, 레지스터 간접, 변위(상대, 인덱스, 베이스) 방식이 있다.
직접 주소 지정	EA=A로 한 번의 메모리 접근으로 데이터 인출 가능하나 주소 비트가 제한되어 범위가 작다.
직접 주소 예제	A=150이면 EA=150의 데이터 1234를 인출하며, 16비트 명령에서 연산코드 5비트면 주소필드 11비트로 2048개 주소, 16비트 단위라 총 4096B 용량을 지정한다.
간접 주소 지정	EA=(A)로 주소가 가리키는 곳에 유효 주소가 있으며 주소 공간은 단어 길이에 의해 확장된다.
간접 주소 장단점	2^n까지 주소 가능해 확장성이 크지만 유효 주소 인출과 데이터 인출로 2번의 메모리 접근이 필요하다.
간접 비트	명령어에 I 비트를 두어 I=0은 직접, I=1은 간접으로 구분한다.
다단계 간접	EA=((..(A)..)) 형태로 여러 단계의 간접을 허용한다.

주제/단락	내용
간접 주소 예제	A=172이면 EA=(172)=202, 데이터는 (202)=3256을 인출하며 16비트 주소로 2^16×2B=128KB를 지정한다.
묵시적 주소지정	데이터 위치가 묵시 지정되며 PUSH R1은 TOS에 R1을 저장, POP은 TOS를 AC로, SHL은 AC를 좌시프트한다.
묵시적 장단점	오퍼랜드 수가 0 또는 1이라 명령어가 짧지만 종류가 제한된다.
즉시 주소 지정	오퍼랜드 필드에 실제 데이터가 포함되어 초기값 설정 등에 쓰이며 메모리 접근이 불필요하나 상수 크기가 필드 비트 수로 제한된다.
레지스터 주소지정	데이터가 레지스터에 있고 오퍼랜드가 그 레지스터를 가리키며 EA=R, 사용 가능 레지스터 수는 2^k이다.
레지스터 주소 장단점	비트 수를 적게 쓰고 메모리 접근 없이 빠르지만 저장 공간이 레지스터로 제한된다.
레지스터 간접 주소	EA=(R)로 레지스터 내용이 주소가 되며 16비트면 64K, 32비트면 4G 주소를 지정할 수 있으나 데이터 인출에 메모리 접근 1회가 필요하다.
레지스터 예제	R=2일 때 레지스터 방식은 R2의 151을 사용하고, 레지스터 간접은 (151)=5678을 사용한다.
변위 주소 지정	EA=A+(R)로 직접과 레지스터 간접의 조합이며 사용 레지스터에 따라 상대, 인덱스, 베이스 방식이 된다.
상대 주소 지정	PC를 사용해 EA=A+(PC)로 계산하며 A는 2의 보수로 표현되고 주로 분기에 쓰인다.
상대 주소 장단점	레지스터 필드 없이 적은 비트로 분기 가능하지만 분기 범위가 오퍼랜드 비트 수로 제한된다.
상대 주소 예제	JUMP가 450번지일 때 A=21이면 PC=451이므로 472로, A=-50이면 401로 분기한다.
인덱스 주소지정	EA=(IX)+A로 배열 접근에 적합하며 자동 인덱싱 시 실행 때마다 IX가 증감된다.
인덱스 예제	배열 시작 500, IX=3이면 네 번째 데이터에 접근한다.
베이스-레지스터 주소	EA=(BR)+A로 프로그램이나 데이터 블록 기준 주소를 담아 위치 이동 시 베이스만 바꿔 재배치를 쉽게 한다.
주소지정 방식 재정리	직접, 간접, 묵시적, 즉시, 레지스터, 레지스터 간접, 변위(상대, 인덱스, 베이스) 방식이 있다.

주제/단락	내용
CISC 개념	복잡 명령어 집합으로 명령 수가 많고 길이가 가변이며 주소지정이 다양해 실행 시간이 길어지나 프로그램 길이는 짧아진다.
RISC 개념	축소 명령어 집합으로 명령 수를 최소화하고 고정 길이, 단순 주소지정을 채택해 프로그램은 길어지지만 단순·규칙적이다.
x86과 ARM 비교	x86은 CISC, 데스크탑·노트북·서버용으로 전력 소비가 높고 복잡 계산과 멀티태스킹에 강하며 강한 하위 호환성을 제공한다.
ARM 특징	RISC 기반으로 모바일·임베디드·IoT에 적합하며 저전력, 효율과 배터리 수명이 강점이고 최신 칩은 고성능을 제공한다.
설계 철학 차이	x86은 복잡 기능을 하드웨어로 지원하고 ARM은 단순 명령으로 소프트웨어 최적화와 모듈식 확장성에 집중한다.
대표 기업	x86은 인텔, AMD 등이 대표적이며 ARM은 애플 실리콘, 삼성 엑시노스, 퀄컴 스냅드래곤 등이 있다.