주제/단락	내용
CPU 역할	기억장치에 저장되어 명령어들을 인출하여 실행함으로써 실제적인 작업을 수행
명령어 사이클 정의	CPU가 한 개의 명령어를 실행하는 데 필요한 전체 처리 과정
명령어 사이클 반복	CPU가 프로그램 실행을 시작한 순간부터 전원을 끄거나 회복 불가능한 오류 발생 시까지 반복
명령어 사이클 부사이클	인출 사이클(fetch cycle)과 실행 사이클(execution cycle) 두 개로 분리
인출 사이클 정의	CPU가 기억장치로부터 명령어를 읽어오는 단계
실행 사이클 정의	명령어를 해독하고 실행하는 단계
기본 명령어 사이클	시작 → 인출 사이클(명령어 인출) → 실행 사이클(명령어 실행) → 중단(또는 반복)
프로그램 카운터(PC)	다음에 인출할 명령어의 주소를 가지고 있는 레지스터
PC 동작 (인출 후)	명령어가 인출된 후 자동적으로 일정 크기(한 명령어 길이)만큼 증가
PC 동작 (분기 시)	분기(branch) 명령어가 실행되는 경우에는 목적지 주소로 갱신
누산기(AC) 역할	데이터를 일시적으로 저장하며, ALU의 산술 연산과 논리 연산 과정에 사용
누산기(AC) 길이	레지스터 길이는 CPU가 한 번에 처리할 수 있는 데이터 비트 수(단어 길이)와 동일
명령어 레지스터(IR)	가장 최근에 인출된 명령어 코드가 저장되어 있는 레지스터
제어 장치 동작 (IR)	제어 장치는 IR에서 명령어를 읽어 와서 해독하고 명령 수행을 위해 각 장치에 제어 신호를 전송
메모리 주소 레지스터(MAR)	PC에 저장된 명령어 주소가 시스템 주소 버스로 출력되기 전에 일시적으로 저장되는 레지스터
메모리 버퍼 레지스터(MBR)	기억장치에 쓰여질 데이터 혹은 읽혀진 데이터를 일시적으로 저장하는 레지스터 (MDR이라고도 함)
데이터 레지스터(DR)	CPU 내의 데이터를 일시적으로 저장하기 위한 범용 레지스터
입출력 주소 레지스터(I/O AR)	특정 I/O 장치의 주소를 지정하는 데 사용
입출력 버퍼 레지스터(I/O BR)	I/O 모듈과 CPU 간에 데이터를 교환하는 데 사용
스택 포인터(SP) 정의	메모리의 한 블록(스택)의 최상위 주소를 저장하는 스택 제어 레지스터
스택 특징	데이터는 후입 선출(LIFO) 방식으로 저장되며, 복귀할 주소 정보를 저장
상태 레지스터(SR) 역할	CPU가 작동하는 동안 특정 조건 발생을 표시하는 데 사용 (특수 목적용)
상태 레지스터 저장 내용	명령어 실행 결과에 따른 조건(조건 플래그)들을 저장

주제/단락	내용
플래그 레지스터	상태 레지스터는 플래그 레지스터(FR) 또는 프로그램 상태 워드(PSW)라고도 함
부호(S) 플래그	직전 산술연산 결과값의 부호 비트 저장 (양수: 0, 음수: 1)
제로(Z) 플래그	연산 결과값이 0이면, 1로 세트
올림수(C) 플래그	덧셈/뺄셈에서 올림수/빌림수가 발생한 경우에 1로 세트
동등(E) 플래그	두 수를 비교한 결과가 같을 경우에 1로 세트
오버플로우(V) 플래그	산술 연산 과정에서 오버플로우가 발생한 경우에 1로 세트
인터럽트(I) 플래그	인터럽트 가능 상태이면 0, 불가능 상태이면 1로 세트
슈퍼바이저(P) 플래그	CPU의 실행 모드가 슈퍼바이저(커널) 모드이면 1, 사용자 모드이면 0으로 세트
클럭 정의	CPU를 비롯한 모든 부품이 일정한 속도로 작동하기 위한 전기적 진동 (펄스)
클럭 발생기	클럭을 만들며, 일반적으로 클럭 수가 클수록 컴퓨터의 처리 속도가 빠름
클럭 단위 (Hz)	1초에 클럭이 몇 번 발생하는지 의미 (예: 1GHz는 1초에 10억 번)
클럭 주기	한 클럭 신호 뒤에서 다음 신호가 올 때까지의 간격
클럭 주기 공식	클럭 주기 = 1 / 클럭 주파수
마이크로 연산 정의	CPU 클럭의 각 주기 동안 수행되는 기본 단위의 동작
인출 사이클 마이크로 연산 (t0)	MAR ← PC (PC가 지정하는 명령어 주소를 MAR로 이동)
인출 사이클 마이크로 연산 (t1)	MBR ← M[MAR], PC ← PC + 1 (MAR 주소 명령어를 MBR에 적재, PC 내용 1 증가)
인출 사이클 마이크로 연산 (t2)	IR ← MBR (MBR의 명령어를 명령어 레지스터 IR로 이동)
실행 사이클 정의	명령어 코드를 해독하고, 그 결과에 따라 필요한 연산들을 수행
CPU 연산 종류 (1)	데이터 이동: CPU와 기억장치 간 혹은 I/O 장치 간에 데이터를 이동
CPU 연산 종류 (2)	데이터 처리: 데이터에 대하여 산술 혹은 논리 연산을 수행
CPU 연산 종류 (3)	데이터 저장: 연산 결과 데이터를 기억장치에 저장
CPU 연산 종류 (4)	프로그램 제어: 프로그램의 실행 순서를 결정
실행 사이클 결정	수행되는 마이크로 연산들은 명령어의 연산 코드(OP code)에 따라 결정됨
연산 코드(OP code)	CPU가 수행할 연산을 지정
오퍼랜드(operand)	명령어 실행에 필요한 데이터가 저장된 주소(addr)
LOAD 명령어 (t0)	MAR ← IR(addr) (IR의 주소 부분을 MAR로 전송)
LOAD 명령어 (t1)	MBR ← M[MAR] (기억장소로부터 데이터를 인출하여 MBR에 전송)
LOAD 명령어 (t2)	AC ← MBR (MBR에 있는 해당 데이터를 AC에 적재)

주제/단락	내용
STORE 명령어 (t0)	MAR ← IR(addr) (데이터를 저장할 기억장치의 주소를 MAR로 전송)
STORE 명령어 (t1)	MBR ← AC (저장할 데이터를 버퍼 레지스터인 MBR로 이동)
STORE 명령어 (t2)	M[MAR] ← MBR (MBR의 내용을 MAR이 지정하는 기억장소에 저장)
ADD 명령어 (t0)	MAR ← IR(addr) (IR의 주소 부분을 MAR로 전송)
ADD 명령어 (t1)	MBR ← M[MAR] (기억장소로부터 데이터를 인출하여 MBR로 전송)
ADD 명령어 (t2)	AC ← AC + MBR (해당 데이터와 AC의 내용을 더하고, 결과값을 다시 AC에 저장)
JUMP 명령어 (t0)	PC ← IR(addr) (명령어의 오퍼랜드(분기할 목적지 주소)를 PC에 저장)
정수 연산 코드	로드(LOAD): 1, STA: 2, ADD: 5, 점프(JUMP): 8
JUMP 170 실행	분기될 목적지 주소(170)가 PC로 적재되어 다음 명령어는 170번지 인출
간접 사이클 위치	인출 사이클과 실행 사이클 사이에 위치
간접 사이클 사용	간접 주소지정 방식(indirect addressing mode)에서 사용
간접 사이클 목적	명령어 주소 필드의 내용으로 데이터의 실제 주소(유효주소)를 인출하여 IR 주소 필드에 저장
간접 사이클 마이크로 연산 (t0)	MAR ← IR(addr) (IR의 주소 필드를 MAR로 전송)
간접 사이클 마이크로 연산 (t1)	MBR ← M[MAR] (MAR 주소로부터 유효주소 정보를 MBR에 적재)
간접 사이클 마이크로 연산 (t2)	IR(addr) ← MBR (MBR의 유효주소 정보를 IR의 주소 필드로 전송)
프로그램 실행 시간 계산 예시	(LOAD 9 + ADD 9 + STORE 6 + JUMP 4) 클럭 주기 = 28 클럭, 실행시간 = 14ns (클럭 주기 0.5ns 가정)
인터럽트 정의	CPU가 프로그램 실행 중에 다른 프로그램 요구로 현재 프로그램을 중단하고 요구된 프로그램을 실행
인터럽트 시스템 기본 요소	인터럽트 요청 신호, 인터럽트 처리 루틴, 인터럽트 서비스 루틴
인터럽트 서비스 루틴 (ISR)	인터럽트를 처리하기 위하여 수행되는 프로그램 루틴
인터럽트 벡터 테이블	다양한 인터럽트 신호를 처리하는 ISR의 시작 주소를 포함
인터럽트 처리 (단계 1)	인터럽트 요청 신호가 발생
인터럽트 처리 (단계 2)	CPU는 현재 명령어 완료 후 프로그램을 일시 중지

주제/단락	내용
인터럽트 처리 (단계 3)	수행 중인 프로그램 상태(PC 내용 등)를 안전한 장소(예: Stack)에 보관
인터럽트 처리 (단계 4)	해당 인터럽트 서비스 루틴의 시작 주소를 PC에 적재하여 실행
인터럽트 처리 (단계 5)	ISR 종료 후 보관했던 레지스터 내용과 PC 내용을 복구하여 중단된 곳부터 다시 수행
인터럽트 사이클 (t0)	MBR ← PC (PC의 내용을 MBR로 전송)
인터럽트 사이클 (t1)	MAR ← SP, PC ← ISR의 시작 주소 (SP 내용을 MAR로, PC 내용을 ISR 시작 주소로 변경)
인터럽트 사이클 (t2)	M[MAR] ← MBR, SP ← SP - 1 (원래 PC 내용을 스택에 저장하고, SP는 1 감소)
다중 인터럽트 정의	인터럽트 서비스 루틴 수행 중에 다른 인터럽트가 발생
다중 인터럽트 처리 [1]	ISR 처리 중 새로운 인터럽트 요구가 들어와도 인터럽트 플래그를 비활성화하여 사이클을 수행하지 않는 방법
다중 인터럽트 처리 [2]	인터럽트의 우선순위 기반으로, 더 높은 우선순위의 인터럽트가 들어오면 현재의 ISR 수행을 중단하고 먼저 처리
인터럽트 우선순위 순서	외부 인터럽트 > 내부 인터럽트 > 소프트웨어 인터럽트
외부 인터럽트	전원, I/O 장치, 타이머 등 컴퓨터 외부 요인으로 발생
내부 인터럽트	잘못된 명령어나 데이터 사용 시, CPU 내부에서 발생 (예: 0으로 나누기, 오버플로우)
소프트웨어 인터럽트	프로그램이 작업 수행을 위해 운영체제의 특정 기능을 요청할 때 발생 (예: SuperVisor Call)
서브루틴 호출 정의	한 블록 실행 중에 다른 블록의 명령어를 삽입 실행하기 위해 CALL과 RET 명령어가 함께 사용
CALL 명령어	현재의 PC 내용(복귀 주소)을 스택에 저장하고 서브루틴의 시작 주소로 분기하는 명령어
RET 명령어	CPU가 원래 실행하던 프로그램으로 복귀(return)시키는 명령어
CALL X 명령어 (t0)	MBR ← PC (현재 PC 내용(복귀 주소)을 MBR로 전송)
CALL X 명령어 (t1)	MAR ← SP, PC ← X (SP 내용을 MAR로, PC를 서브루틴 시작 주소 X로 변경)
CALL X 명령어 (t2)	M[MAR] ← MBR, SP ← SP - 1 (복귀 주소를 스택에 저장하고, SP는 1 감소)
RET 명령어 (t0)	SP ← SP + 1 (스택 포인터 1 증가)
RET 명령어 (t1)	MAR ← SP (SP 내용을 MAR로 전송)
RET 명령어 (t2)	MBR ← M[MAR] (스택에 저장된 복귀 주소를 MBR로 인출)

주제/단락	내용
RET 명령어 (t3)	PC ← MBR (MBR의 복귀 주소를 PC에 적재)
저장 파일 이름	2. CPU의 구조와 기능2.md