마이크로컴퓨터 구조 개요 (파랑이 시험, 빨강 쉽시간에중요하다한거)

하드웨어 : 핸드폰의 몸체, 메인보드, 컴퓨터의 부품 등 딱딱하고 물리적으로 형태가 있다.
컴퓨터의 중앙처리장치, 기억장치, 입출력장치(마우스,키보드,모니터) 등 전자, 기계 장치의 몸체
소프트웨어 : 형태가 없고 논리적인 것. 컴퓨터에서 운용되는 프로그램이나 문서

-전기회로의 역사
1. 진공관의 등장 : analog era
2. 반도체의 등장
3. 트랜지스터 개발 : 증폭(아날로그)와 스위칭(디지털)
4. 스위칭 & 2진수 : digital era
5. 이진연산을 위한 디지털회로 - 기본적인 회로 : and, or, xor, not
   조합 회로 : decoder, mux, adder
   순차 회로 : 래치, 플립플롭, 카운터, 레지스터
6. 디지털회로의 한계 : Hard & Non-Flexible -> 수정, 고치기 힘듦
7. 마이크로 프로세서의 등장 : Soft & Flexible -> 기능을 수정하기 쉬움 -> 컴퓨터시대

펌웨어 : 하드웨어와 소프트웨어의 중간 형태로 하드웨어 화 된 소프트웨어이다.

ALU : CPU내의 데이터 연산 (산술연산, 논리연산, 비트연산)을 수행하는 핵심요소.

-일반디지털회로와 마이크로프로세서회로(CPU)의 도입
일반디지털회로 : 회로의 기능을 바꾸려면 하드웨어 조작이 필요.
한 가지 기능(ONLY H/W), 시간표시 변경의 한계
마이크로프로세서회로 : 회로의 기능을 소프트웨어만으로 변경 가능.
무한 기능(H/W + S/W), 시간표시 자유로운 변경 가능

컴퓨터 : 소프트웨어를 통해서 하드웨어를 변형시키는 전자 기기
컴퓨터의 3가지 구성요소 : CPU, Memory, I/O Peripherals
CPU(central processing unit) : 여러 명령어들로 이루어진 프로그램내의 명령어들을 하나씩 읽어 해독하고 명령에 해당하는 컴퓨터의 각 요소들을 제어하는 신호를 발생시키는 컴퓨터의 핵심 모듈
프로세서 : CPU
마이크로프로세서 : 원칩으로 만들어진 CPU
마이크로컴퓨터 : 마이크로프로세서로 구성된 컴퓨터
MCU(micro controller unit) : 임베디드 제어를 목적으로 제조된 원칩 마이크로컴퓨터

컴파일러 : C언어 문장(고급 프로그래밍 언어)을 기계어 문장으로 변환하는 프로그램.
- self compiler : c파일이 존재하는 컴퓨터의 CPU가 인식 가능한 기계어 파일로 변환하는 컴파일러 (Visual studio 내장 컴파일러, PC에서 실행하는 파일을 생성하는 컴파일러)
- Cross compiler : 타 컴퓨터의 cpu가 인식 가능한 기계어 파일로 변환하는 컴파일러. (IAR workbench 내장 컴파일러, 모든 임베디드 컴퓨터용 실행파일을 생성하는 컴파일러)

기계어 : CPU가 인식하는 이진수 조합으로 된 코드 (8,16,32,64 bit)

IDE(Integrated Development Environment) : 비주얼 스튜디오 내에 컴파일러, 링커, 에디터, 디버거 등 여러 가지 프로그램이 통합되어 있는 것. (통합 개발 환경)

기계어 파일의 저장매체 : RAM, ROM, Flash (반도체 메모리)
- pc의 경우 : HD/SSD에 저장했다가 실행될 때 DRAM에 load된 후 실행됨.
- 임베디드 컴퓨터의 경우 : Flash/ROM에 저장되었다가 바로 실행됨.

메모리의 특성
-기본단위 : 1-byte
-기본저장단위 : 8, 16, 32, 64 bit
- 각 저장단위 마다 주소가 배정되어 관리.

1. CPU는 읽어야할 기계어코드가 있는 주소를 메모리에 전송하고 메모리는 주소를 받고 해당 주소의 기계어코드를 CPU에 전송.
2. CPU내부의 해독기에서 기계어코드를 해독하고 해독된 명령어에 따라 실행해야할 동작을 위해 CPU는 순차적으로 제어신호를 컴퓨터 내부의 각 모듈로 전송.
(CPU내부의 Controller : Instruction Decoder + Signal Generator/Sequencer)

컴퓨터의 하드웨어 요소
- 레지스터 : 데이터를 저장하는 기본소자
- ALU : CPU내에 위치하고 연산을 수행
- Controller : CPU내에 위치하고 명령어를 해독하여 컴퓨터를 동작시키는 제어신호발생.
- Memory : 데이터와 프로그램을 저장.
- I/O peripherals : 사용자나 다른 컴퓨터와의 인터페이스를 위한 장치

컴퓨터의 소프트웨어 요소
- 데이터 : 컴퓨터 동작의 대상
- 프로그램 : 컴퓨터 동작을 실현

마이크로컴퓨터의 구조와 동작

CPU : 사용자가 입력한 명령어(프로그램)를 메모리로부터 읽어(Fetch), 해독하고 동작을 실행(Execute)하는 컴퓨터의 핵심모듈.

마이크로컴퓨터 : 마이크로프로세서에 메모리장치가 연결되고 또 외부와 데이터를 송수신 할 수 있는 장치(I/O port)가 장착된 장치.

원칩 마이크로컴퓨터 : 원칩으로 된 마이크로컴퓨터.

MCU : 임베디드 제어를 목적으로 제조된 원칩 마이크로컴퓨터. (Cortex-M4)
- CPU core, 메모리, 입/출력모듈을 가지고 있다.
- NOR Flash메모리 기계어 코드를 써 넣음. (프로그램 메모리)
- 데이터 저장을 위해 SRAM을 가지고 있다. (데이터 메모리)
- PC가 다양한 요구에 따라 동작하는 일반적인 일에 사용된다면 MCU는 기능을 설정하고 정해진 일을 수행하도록 프로그래밍 되어 장치 등에 장착되어 동작한다.

임베디드시스템 : 제어를 위한 특정 기능을 수행하는 컴퓨터 시스템으로 장치 내에 존재하는 전자 시스템. 대부분 MCU나 DSP를 메인 프로세서로 장착.

마이크로프로세서(CPU)의 3가지 구성 요소
- ALU : 산술연산과 논리연산을 수행하는 장치
- 레지스터 : 마이크로프로세서에서 프로그램이 수행되는 동안에 데이터를 일시적으로 저장.
- 제어장치 : 명령어를 해독하여 컴퓨터를 동작시키는 각종 제어신호를 발생시키는 장치.

마이크로프로세서 버스
-전원선
-정보선 : 단선 - 1bit의 독립적인 정보를 전달하는 선
버스 - 같은 성질을 갖는 정보들을 묶어서 함께 송수신하는 선
--> 여러 데이터를 동시에 전송, 공통으로 사용 (Multiple, common)

버스
- 데이터 버스 : 데이터나 기계어코드가 마이크로프로세서 내부로 들어오거나 혹은 명령 수행이나 계산의 결과를 마이크로프로세서로부터 밖으로 내보낼 때. (양방향성 버스)
(메모리 수에 비례하여 마이크로프로세서의 핀 증가 -> 버스개념: 모든 메모리가 같은 데이터를 저장하거나 데이터 충돌 -> chip enable : 메모리 수에 비례하여 마이크로프로세서의 chip enable핀이 증가 -> 어드레스버스 도입)
- 어드레스 버스 : 마이크로프로세서가 주소 코드를 메모리 혹은 다른 외부장치로 한 방향으로 보낼 때 사용되는 버스. (어드레스 비트수가 클수록 메모리 영역도 증가)
- 제어 버스 : read or write 방향 설정

마이크로프로세서 프로그래밍
: 마이크로프로세서를 동작하기 위한 일련의 명령어를 작성하고 마이크로프로세서에 제공하는 작업.
- 고급 프로그래밍 언어 : c, java, 파이썬
- 어셈블리언어 : 마이크로프로세서를 직접 제어하는데 쓰이는 2진 코드
-> 어셈블러
- 기계어 : 마이크로프로세서만이 인식하는 2진 코드로 표현된 명령어.

컴파일러 : 고급 프로그래밍 언어를 기계어로 번역하는 프로그램
->어셈블리언어, 기계어는 마이크로프로세서마다 다름
->고급 프로그래밍 언어는 마이크로프로세서에 무관

마이크로컴퓨터 = 마이크로프로세서(CPU) + 기억장치 + 입출력장치

마이크로프로세서가 버스를 통해 기억장치 및 입출력장치와 정보와 신호를 교환.
주소는 어드레스버스를 통해 메모리나 입출력장치에 보내지고 명령 또는 데이터는
데이터 버스를 통해서 CPU와 기억장치 및 입출력 장치 사이에 전송.

BIOS : 컴퓨터의 가장 기본적인 기능을 처리해 주는 프로그램들의 집합.
(운영체제에서 I/O 주변장치를 구동하기 위한 루틴들의 집합체로서 운영체제 가장 하위에 있는 부분으로 다른 S/W는 모두 이 층을 기반으로 하여 실행된다.)
(하드웨어와 소프트웨어 사이의 연결과 번역 기능을 담당하는 인터페이스)
- 스타트업 루틴 : 컴퓨터가 켜질 때 자동으로 실행되어 컴퓨터의 상태를 검사하고 시스템을 초기화하는 작업을 한다.(어떤 주변장치가 연결되어 있는지 확인)
- 서비스 처리루틴 : 사용자 프로그램 또는 OS가 요구하는 일을 처리.

중앙처리장치(CPU) : CPU는 메모리 주소를 지정하고 거기에 저장된 명령어를 가져와서 (Fetch) 그 명령을 실행(execute)
한 명령어를 완료한 후에 CPU는 다음 명령어로 이동하며 하나의 프로그램 내에 있는 모든 명령어들이 완전히 처리될 때 까지 페치와 실행과정을 반복.
(모든 동작은 CPU 외부에 부착된 오실레이터(클락)에 동기 되어 이루어진다.)
(클락 -> 트리거신호를 주기 위하여, 동기화를 하기 위하여)

- 8086 : 16/20, 10M Hz
(데이터 버스 크기 / 어드레스 버스 크기) , 클럭주파수
- 8088 : 16(8)/20, 12.5M Hz
(내부에서는 16bit, 외부에서는 8bit)
10MHz일 때 한번 read하는데 걸리는 시간은? - 타이밍도그림중요

CPU 성능(속도)를 결정하는 요소들
- 클락 주파수 (가장 큰 요인)
- 데이터 버스 사이즈 : 8, 16, 32, 64 bit
- 캐시 사이즈 (cpu의 비서 역할)
- Command structure : CISC(복잡), RISC(간단)
- Pipelinling
- Parallel processing : dual-core, quad-core (core수가 많을수록 빠른 연산)

- 기본 동작
1. 패치 (operator 읽어옴)
2. read operand
3. Execute
4. Write result (결과물 저장)

마이크로프로세서 핀 구성 -> power, clock, reset, data, address, controller bus

8086/8088의 특징
1. 2개의 분리된 내부 장치에 의해 수행(EU, BIU)
   BIU : instruction queue, program counter
   EU : SP, Flags
2. Pipelining 구조 : Prefetching, Instruction queue
 -> cpu내부가 여러부분으로 나뉘어 동시에 실행될 수 있게 해주는 구조

특수 레지스터 : CPU가 전용으로 사용하는 CPU내 특수용도 메모리
1. PC(프로그램 카운터) : 다음에 읽어 들일 명령어의 위치를 저장하는 특수 레지스터.
2. Status Reg(Flag Reg) : 연산결과의 부호, 자리올림여부의 정보를 가지는 특수레지스터.
3. Stack Pointer : LIFO형태로 운영하는 stack메모리 공간을 활용할 때 마지막에 입력된 데이터를 가리키도록 사용하는 레지스터.

CPU가 메모리에 주소를 보내면 메모리에서는 해당 주소부분이 반응을 하고 read신호가 들어와야지만 데이터가 읽어져서 cpu로 보내진다. (read신호를 받는 것이 중요)

8088 마이크로프로세서 (16(8)/20)
- 외부 어드레스버스와 데이터버스가 결합된 형태 : multiplexing
(핀 한 개가 두 가지 이상의 기능을 하는 대신에 동시에 실행될 순 없으므로
bus controller와 address latches를 사용한다,)
-address latch : address bus와 data bus가 충돌되는 것을 예방.
(address 신호가 먼저 발생하면 address latch가 ALE 신호를 받아 어드레스를 통과하고 데이터 신호가 발생하기 전에 disable하여 latch입구를 막는다.
-bus transceiver : data bus 방향을 제어한다.
(read시 Receive , write시 Transmit)
-bus controller : address latches를 ALE 신호를 발생시킴.
메모리나 I/o의 read/write 신호를 발생
데이터 버스 제어 신호 발생

주소에 관한 규칙
1. 컴퓨터내의 데이터버스에 연결된 모든 메모리와 IO Port들은 주소를 할당받아야 한다.
2. 배정은 컴퓨터 H/W 설계자가 한다.
3. 주소는 중복되면 안된다.
4. 컴퓨터 내의 주소배정은 Address Decoder에 의해서 수행된다.

Address Decoder : CPU가 발생하는 주소에 해당하는 메모리나 I/O포트를 ENABLE.

I/O 포트 (프린트에 나온 그림이 주어지고 이 그림이 무슨 방식인지 이름 쓰는 문제)
1. I/o mapped I/o(전용주소 I/O 주소지정) : i/o 전용 명령어를 사용 -> 8088, 범용컴퓨터
(주소가 중복가능.)
(주소를 2배 할당 가능./선이 복잡하다.)
2. Memory mapped I/O(메모리-맵 I/O 주소지정) : i/o와 메모리가 같은 명령어를 사용 -> 임베디드
(cortex-M4, MCU)
(주소가 중복불가)
(신호선의 관리가 편해진다.)

I/O인터럽트 (임의의 시간에 임의의 작업)
: 주변장치가 CPU에 작업을 요청할 때 사용하는 방법.
-Polling : cpu가 I/o에게 계속 물어보는 것. -> 시간낭비, 우선순위 문제
-Interrupt : I/o가 cpu에게 요청하는 것. -> PIC 우선순위에 입각하여 인터럽트 처리.
I/O가 CPU에게 신호를 주었을 때(Request) CPU가 그 정보를 읽어오는 것(Service).
(인터럽트 신호는 버스선이 아닌 request line을 통해 request하고 cpu가 그 킷값을 데이터 버스선을 통해 읽어들인다.)

메모리

메모리 : 오랜기간 및 짧은 기간동안의 2진데이터를 저장하는 장치
1 cell = 1 bit 저장
반도체 메모리 : 프로그램(기계어코드)저장, 데이터 저장

정보의 저장 단위 : word - 1bit, 1byte, 2byte, 4byte (보통은 2byte)
(1 word = 1 address)
on-system에서 write가능한 것을 고르시오. 휘발성을 모두 고르시오.
ROM (Read Only Memory) : on-system에서 read만 가능한 memory.
(칩을 꺼내서 칩에 정보를 write해주고 보드에 꽂아서 read 해야 함.)
- writing은 off-system에서만 가능 (EEPROM은 예외)
- 불휘발성 메모리 : power off시 저장된 데이터가 유지됨
- Mask ROM : 제조 시 데이터를 writing. 구매자는 writing 못함.
- PROM : 한번만 writing 수 있는 ROM
- EPROM : 여러 번 지웠다가 writing 할 수 있는 ROM
  UV EPROM - 전기적 write, 광학적(자외선) erase -> off-system writing
  EEPROM - 전기적인 펄스로 write, erase -> on-system writing
  (용량을 크게 만들지 못한다.)
  (메모리를 지운다는 것은 1로 만든다는 것이다.)

3-state 버퍼 기능과 버스 (왜 필요하고 어떻게 동작하는지)
왜 필요? 3-state 버퍼 기능이 없으면 모든 메모리에 같은 데이터가 입력되거나 데이터 간에 충돌이 일어 날 수 있기 때문에 high-Z상태를 이용하여 데이터의 입출력을 관리해 주어야 한다

어떻게 동작하는지 ? enable신호가 high일때는 데이터의 입출력이 자유롭게 이루어 지며
enable신호가 low일때는 게이트가 고저항상태로 high -Z가
되어 데이터의 입출력이 이루어지지 않게 문을 닫는다.

RAM (Random Access Memory) : on-system에서 read 및 write가 가능한 memory
- 휘발성 메모리 :power off시 저장된 데이터가 사라짐
- 읽고 쓰는데 시간적. 공간적 제한이 없다.
- SRAM(Static) (안정적) : 셀 1개 = 플립플롭 1개
  임베디드 컴퓨터
- DRAM(Dynamic)(불안정) : 셀 1개 = 1-TR, 1-CAP -> 집적도가 매우 높다.
  PC(범용 컴퓨터)
  cap의 충방전으로 인한 1-bit 저장
  오래 두면 방전하므로 refresh 필요.

플래쉬 메모리 : 비휘발성인 고밀도의 READ/WRITE 메모리 (on-system)
floating-gate MOS TR을 이용하여 충전 및 방전
재충전 필요 없음.
NOR type : 프로그램 저장용
NAND type : 데이터 저장용(자주 바뀌는 메모리)

FIFO (first in -first out) : Queue
구조 : Standalone-type Queue
- queue는 RAM에 설정하지 않고 독립적인 모듈로 제작.
- Instruction 처리, 컴퓨터 내의 process의 순서 관리.

LIFO (last in - first out) : Stack
- stack은 보통 RAM의 끝부분에 설정.
- Subroutine call 시 return address 저장
- PUSH 동작 : stack pointer 감소 -> 데이터 저장 (data in)
- POP 동작 : stack pointer 증가 -> 데이터 read

기억형태	비휘발성	고집적도	하나의 TR cell	시스템내에서 쓰기 가능 여부
Flash	O	O	O	O
SRAM				O
DRAM		O	O	O
ROM	O	O	O	
EPROM	O	O	O	
EEPROM	O			O

NOR 플래쉬 메모리	프로그램 저장용
NAND 플래쉬 메모리	데이터 저장용
DRAM	데이터/변수 저장용 (범용컴퓨터)
SRAM	데이터/변수 저장용 (임베디드 컴퓨터)
EPROM	프로그램 저장용
EEPROM	작은 데이터 저장용

Cprtex-M4 (STM32F407)

MCU(Micro Controller Unit) : 칩 1개로 이루어진 마이크로 컴퓨터

마이크로프로세서와 MCU 분류

1. 프로그램을 access하는 버스구조에 따른 분류
   - 폰 노이만 구조 : 범용컴퓨터
   - 하바드 구조 : 임베디드 컴퓨터
     Instruction bus (program memory)
     + Data bus (data memory)
     (프로그램전용 address bus와 데이터 전용 address bus 존재)
2. 명령어의 다양성 기준에 따른 분류
   - CISC(complex instruction set computer)
     : 명령어의 종류와 형태가 다양, 호환성 우수 -> CPU의 구조가 복잡
     대부분의 데스크탑 PC에 사용되는 프로세서
   - RISC(reduced instruction set computer)
     : 명령어의 수를 줄이고 CPU의 구조를 단순화 -> 빠른 처리 속도
     (꼭 필요한 명령어로만 처리되어있다.)
     임베디드 시스템에 많이 사용된다.

MCU의 특징 : Register based Control & Monitoring
(mcu내부의 모든 요소들의 제어나 모니터링은 해당 I/o레지스터를 통해서만 가능하다.)

ARM (Cortex-M4) : 하바드 구조+RISC+32-bit (4G의 주소 생성)+thumb2 mode

주요 Cortex-M 시리즈 종류 및 특성

항목	M3	M4	M7
동작 속도(DMIPS/MHz)	1.25	1.25	2.14
명령어/데이터 버스	32	32	64
pipe line	3	3	6
DSP	없음	지원	지원
FPU	없음	옵션 (Single precision)	옵션 (Single/Double precision)
MPU	0~8	0~8	0~16
Bit banding	지원	지원	없음

DMIPS (Dhrystone Million Instruction per Second)
: 1MHz클럭에서 1초간 실행되는 명령어 수.

Cortex-M processor의 instruction set (instruction set : 기계어를 표현하는 형식(문법))
- ARM mode : 32bit instruction (속도 면에서 유리) (코드 사이즈 증가)
- Thumb mode : 16bit instruction (프로그램 size절감, 전력절감)
- Thumb-2 mode : 16bit / 32bit instruction (양쪽의 단점 보완)
(같은 c언어로 된 명령들을 다른 mode로 컴파일하면 다른 기계어 파일 생성)

thumb2 mode : thumb mode의 단점(부족한 명령어)을 보완하기 위해 ARM mode의 장점(다양한 명령어 종류)을 도입한 mode.

DSP : (Digital Signal Process)
덧셈, 뺄셈, 곱셈 등의 반복 연산을 고속으로 처리 할 수 있는 회로.
(하드웨어적으로 연산 전용 마이크로프로세서)
(복잡한 연산기능을 빨리 할 수 있음.)

Cortex-M4 : Cortex-M3 + DSP, FPU
- 32bit core ,1.25 DMIPS/MHz, 3-stage pipeline, 하버드 구조(데이터버스와 명령버스가 분리)
- Thumb 및 Thumb-2 명령어 사용 가능
- 32비트 하드웨어 곱셈, 나눗셈
- Saturated math 지원(계산 후 overflow되면 최댓값으로 대체)
- 1~240개의 인터럽트 처리.
- 슬립모드 : 전력절감 -> clock을 주지 않는다.

cortex-M4 내부 장치들
-NVIC (Nested Vectored Interrupt Controller)
: 인터럽트 제어기이며 인터럽트 처리속도를 지원함.
-MPU (Memory Protection Unit)
: 메모리 영역에 관한 속성을 정의하여 시스템의 신뢰성을 향상시키는 장치.
-bus matrix

Cortex-M4 프로세서 코어
- ALU, Register, Controller
- 레지스터
: Stack Pointer(SP) - MSP : default SP로서 예외 핸들러가 사용
PSP : 사용자 프로그램이 사용
Link register - 서브루틴 호출 시 return address 저장
(하나의 주소만 저장할수있음. ->바로전주소)
Program Counter(PC) - 현재수행중인 프로그램의 주소 저장
프로그램 상태레지스터 - ALU상태, 프로그램 실행상태,
현재 수행중인 인터럽트 번호 등 저장

Cortex-M4 프로세서 코어 : 실행모드
- 쓰레드 모드(일반) : 일반적인 프로그램 실행 시 모드 - 특권, 사용자 레벨
- 핸들러 모드(특별) : 예외(인터럽트) 핸들러 실행 시 모드 - 특권 레벨

프로그램에서 메모리의 접근에 대한 허용수준 종류 (그림 그대로 나옴 - 빈칸뚫어서)
- 특권(privileged) 레벨 : 모든 메모리에 접근이 가능
- 사용자(user) 레벨 : 지정된 메모리에 접근이 가능함.

MPU : 메모리에 대한 접근 권한을 제어하는 모듈
-> 신뢰성 있는 시스템 제작

Cortex-M4 프로세서 코어 : 3단 Pipeline
- 프로세서의 동작 속도를 높임
- 명령어의 처리를 여러 단계로 나누어 실행하는 구조
- 하나의 명령어는 인출(fetch), 해독(decode), 실행(execute)로 나누어 수행
- 병렬처리효과 : 1사이클 당 1명령어 수행

버스 매트릭스
: cortex 프로세서의 내부버스와 외부버스를 상황에 맞게 연결해주는 모듈.
- I-code bus : 코드영역메모리(Flash)에서 명령어를 인출하는 버스
(AHB-Lite 프로토콜을 따르는 32비트 버스)
- D-code bus : 코드영역메모리(Flash)에서 데이터코드를 입출력하는 버스
(AHB-Lite 프로토콜을 따르는 32비트 버스)
- System bus : SRAM영역, 주변장치영역의 메모리에서 명령어인출, 데이터 입출력
(AHB-Lite 프로토콜을 따르는 32비트 버스)

AHB(Advanced High-performance Bus) - 프로세서에 사용되는 버스 프로토콜
APB(Advanced Peropheral Bus) - 프로세서와 주변장치와의 데이터 통신을 위한 버스
프로토콜 (GPIO, 타이머, 인터럽트, UART)

Cortex-M4 프로세서 코어 : 메모리 맵
code -> SRAM -> 주변장치 : 각각 0.5GB씩 할당되어 있다.

code 영역 - 프로그램 코드가 저장되는 영역 (일반적인 경우)
(SRAM영역, RAM영역 - 디버깅 경우)
I-code, D-code 버스를 통해 명령어 인출과 데이터 액세스가 동시에.
CORTEX-m4와 I-code, D-code bus를 통하여 연결
SRAM 영역 - cortex-M4기반의 MCU의 내부에 있는 SRAM을 위한 영역
CORTEX-m4와 I-code, D-code, system bus를 통하여 연결
주변장치 - cortex-M4기반의 MCU의 내부에 있는 주변장치를 위한 영역
CORTEX-m4와 system bus를 통하여 연결

cortex-M4 프로세서 코어 : 인터럽트 및 예외
예외 = 시스템 예외 + 인터럽트 예외
- 내부의 NVIC(중첩 벡터형 인터럽트 제어기)가 각종 인터럽트의 제어 수행
- 시스템 예외 : 프로세서 내부에서 발생
Reset, NMI, Hard Fault
- 외부 인터럽트 : 그 외의 주변장치에서 발생

NVIC : 모든 인터럽트(예외)에 대한 우선순위를 결정하고 처리. (벡터테이블)

스택킹 -> 벡터 인출 -> 레지스터 업데이트
-> 인터럽트 서비스 루틴 실행 -> 인터럽트 종료/빠져나오기.
- 중첩 인터럽트(Nested interrupt) : 우선순위에 따라 높은 순위의 인터럽트를 먼저 처리.
- 벡터형 인터럽트(vectored interrupt) : 벡터 테이블을 이용하여 ISR의 시작주소 알수있음.

////////////////////////////////******지혁 정리***////////////////////////////
마이컴 개요 x
구동동작
MCU 정의 : 임베디드 제어를 목적으로 제조된 원칩 마이크로컴퓨터
,마이크로프러세서 정의 : 원칩으로 된 CPU
버스크기 비트수가 뭘 나타내느냐,
버스 : 같은 성질을 갖는 정보들을 묶어서 함께 송수신하는 선
여러 데이터의 동시전송, 공통으로 사용 의 의미를 가지고 있다.
제어장치(Control unit) : instruction decoder & control signal generator/sequencer
기계어,
컴파일러 : 고급프로그래밍 언어를 기계어로 번역하는 프로그램
CPU : CPU는 메모리에 저장된 명령어를 가져와(Fetch) 그 명령을 실행(decode & control signal generate)한다.
8086(1978) : 데이터 버스-16비트 어드레스버스-20비트, 클락 : 10MHZ
8088 : 데이터 버스-내부 16비트, 외부 8비트(호환성 때문에) 어드레스버스-20비트, 클락 : 12.5MHZ
1.2x
1.3 4가지 기본동작 : 1. Fetch instruction 2. Read operand 3. Execute 4. Write result
파이프라이닝 의미 : 단계별로 수행하던걸 병행해서 처리함
EU,BIU 파이프라인 Q
특수 레지스터 PC 등등 역할
Status Reg(Flag Reg) : 연산결과가 양수인지, 음수인지, 자리올림이 발생했는지 등의 정보를 가지고 있는 레지스터이다.

SP(Stack Pointer) : 스택을 사용할 때 가장 나중에 입력된 데이터의 주소를 저장하는 레지스터다.
PC(Program Counter) : 다음에 읽어들일 명령어의 주소를 저장하는 레지스터다.
레치,버스 트렌시버 역할 반듯이 알 것
래치 : Address 버스 신호가 먼저 발생하면 ALE 신호를 받아 어드레스를 저장하고, 데이터 신호가 발생하기 전에 ALE 신호를 Disable 하여 래치 입구를 차단한다.
버스 트랜시버 : 데이터 Read시 데이터 방향 Receive, Write 시 데이터 방향 Transmit으로 설정
메모리 write 동작 번호 쓰기(4개중에 1개)
메모리 라이트 사이클 그림 이해하기
어드레스 디코더 과제 정도 수준 문제

메모리
메모리 종류들 특징 알고있기
메모리 비교 표 잘 알고있기,분류할수있기
q stack 사용 예 같은 문제 냄

m4
디테일한거는 내기 힘듬
폰노이만 하바드 차이점 알기
섬모드 섬투?
실행모드
비트밴딩x
3.4.8 보기
stm
5.features 빨간부분
쥐피아이오 포트수 등등
부팅 모드 3가지
저전력 모드 클락 등등 잘보기
GPIO
3.H/W 사양 및 특징 페이지 용어,의미