마이크로프로세서 실습
(Micro-processor Practice)

2023년 2학기

이 형 봉
컴퓨터공학과
강릉원주대학교

강의계획

학습목표

- 일반적인 컴퓨터 구조는 프로세서(CPU)와 메모리를 중심으로 기타 주변장치들로 이루어진다. 프로세서는 메모리에 저장된 기계명령어를 읽어 대응되는 연산을 처리하면서, 외부에서 발생하는 사건(인터럽트)를 인식하여 이에 대응하기 위한 처리 루틴으로 점프하는 기본 기능을 갖는다. 이 교과에서는 메모리와 입출력 장치가 On-chip 형태로 일체화된 MCU(Micro Controller Unit)의 내부를 조명하고 활용하는 방법론을 다루고, 구체적인 학습 목표는 다음과 같다.
  - 일반적인 마이크로 프로세서 구조 이해(CPU, 플래시 메모리, 부트로더, 주변 장치)
  - 마이크로 프로세서 소프트웨어 개발 방법론(크로스 컴파일러, 개발도구)
  - 마이크로 프로세서 디바이스 활용(UART, ADC, Timer, SPI, i2C, TWI)
  - 마이크로 프로세서의 IOT 구성 (웹 서버 설정 및 CGI 프로그래밍)

교과내용

- VR ATmega2560 MCU를 대상으로, 부트로더, UART 통신, 타이머, 소프트웨어 타이머, 타스크 개체를 구현하고, Memory Mapped IO와 Isolated IO의 개념을 실험하며, ADC, SPI, I2C 등 각종 센서 프로토콜 활용 및 측정 방법 등을 다룬다. 마지막으로 구현된 MCU 펌웨어를 이용하여 IOT를 구성한다. MCU 소프트웨어 개발은 윈도우 운영체제 하의 AVRStudio 4에서 단계별로 진행하고, IOT를 위한 CGI 프로그래밍은 Visual Stuidio에서 진행한다.

선수교과

- 컴퓨터프로그래밍I, II", "웹프로그래밍II", "유닉스/리눅스 시스템", "시스템프로그래밍", "운영체제", "컴퓨터구조"

수업운영

- 학기중 단계적으로 진행하는 C언어 프로젝트를 구현한다.

평가방법

- 중간고사 : 30%, 기말고사 : 30%, 프로젝트 : 30, 출석 : 10%

교재, 기타

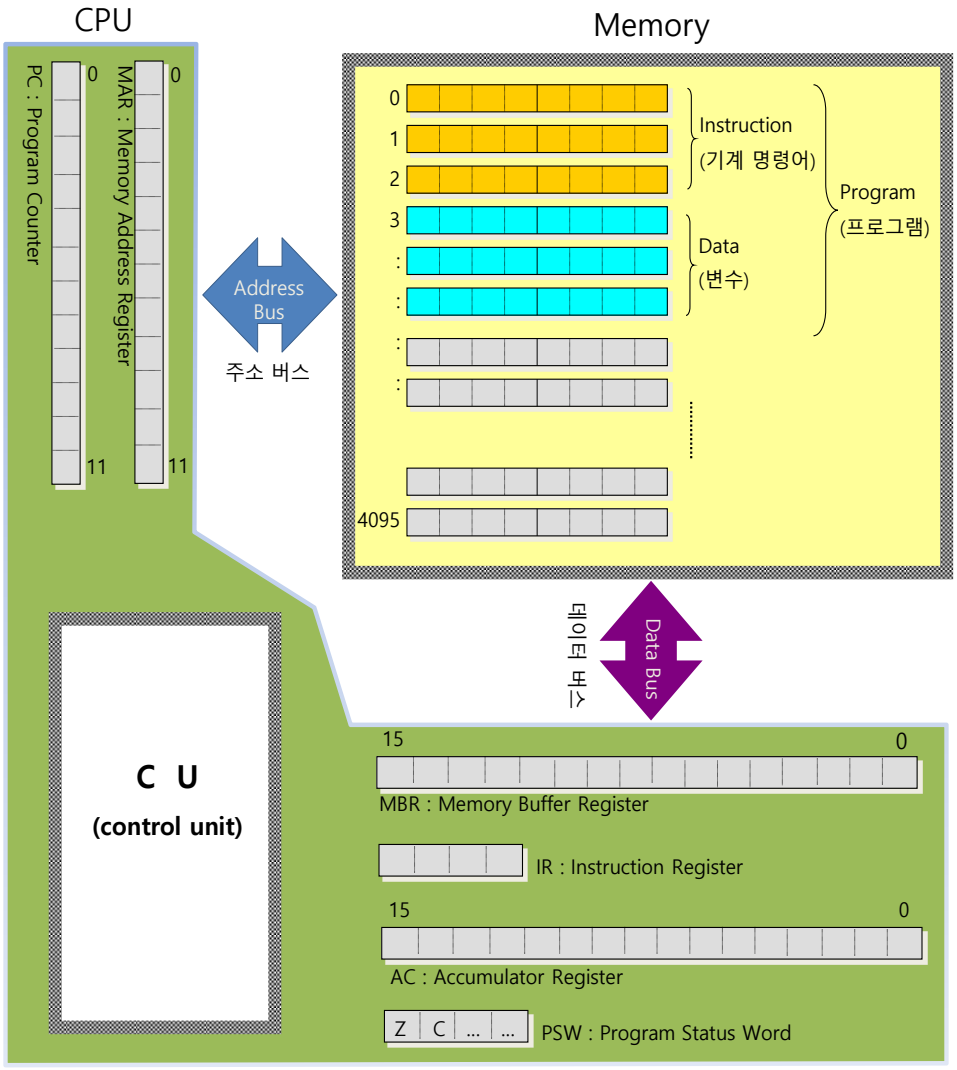
- **교 재**
  - 이형봉 저, "마이크로 프로세서 실습 I (개정판) (GEMS-CRC 모트, Atmega2560)", 도서출판홍릉
  - Atmega2560 Datasheet
  - 주어진 프로젝트와 관련된 각종 기술서적 및 참고문헌

- **기타**
  - 공결은 학과사무실 발급 서류로만 인정
  - 수업 중 코로나 방역수칙 철저
  - 개인별 위생 관리 철저
  - 질문은 가급적 수업 시간에만(참여자 공유)
  - 긴급 사항은 전화(010-5201-2713) 혹은 이메일(hblee@gwnu.ac.kr)로(강의지원 시스템 사용하지 않음)
  - ppt 강의 전용 자료는 배포호지 않음(교과서를 읽고 학습 요망)

0. 교과 배경 지식...

◆ 컴퓨터 시스템 주요 구성 요소

- 중앙처리장치
  (CPU: Central Processing Unit)
- 주기억장치
  (Main Memory)
- 주변 장치
  (Peripheral Device)



0. 교과 배경 지식...

0.1.1 시스템 버스(System Bus)

- **어드레스 버스(Address Bus)**
  - 메모리나 입출력 장치에 접근할 때 접근 장소를 식별하기 위한 주소를 전달
  - 메모리 크기에 따라 8/16/24/32/64 비트(라인) 등의 크기를 가짐
- **데이터 버스(Data Bus)**
  - 메모리나 입출력 장치에(서) 데이터를 보내거나 읽어올 데이터를 전달
  - 데이터 전송 대역폭에 따라 8/16/32/64 비트(라인) 등의 크기를 가짐
- **제어 버스(Control Bus)**
  - 메모리나 입출력 장치 등에 읽기/쓰기 등의 기능을 제어하기 위한 신호 전달
  - 신호의 종류에 1/2/3 비트(라인) 등의 크기를 가짐

0. 교과 배경 지식...

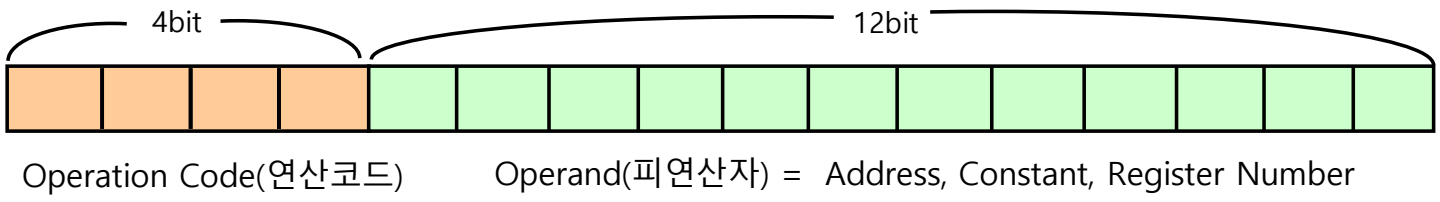
0.1.2 주기억장치(Main Memory)

- 비트(Bit)와 바이트(Byte)
  - 비트 : 0/1을 기록하는 메모리의 최소 구성 소자
  - 바이트 : 8 비트로 구성되어 256 가지의 서로 다른 정보 저장 가능
- 바이트 별 주소 지정
  - 대부분의 컴퓨터는 바이트 단위로 주소를 부여
- 바이트 단위 저장
  - 읽거나 쓰기는 반드시 바이트 단위로 이루어짐
- 1/2/4/8 바이트 동시 접근(읽기/쓰기)
  - 기계 명령어(Machine Instruction)에 따라 동시 접근하는 바이트 수가 다름
- 메모리 얼라이먼트(Memory Alignment)
  - 2/4/8 등의 여러 바이트 동시 접근시 목표 시작 주소를 2/4/8의 배수가 되도록 요구하는 기계(CPU)가 있음

0. 교과 배경 지식...

0.1.3 기계 명령어(Machine Instruction)

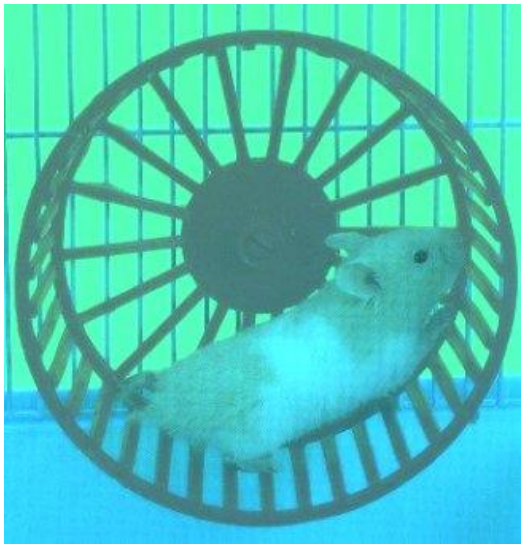
- CPU가 처리하는 명령어 단위
  - 고급 언어로부터 번역되어 실행 시 메모리에 적재
  - CPU에 의해 인출되어 처리됨
- 연산 코드(Operation Code)와 피연산자(Operand)
  - 연산 코드는 피연산자에 적용할 덧셈, 뺄셈 등의 연산을 의미하는 코드
    예: 누산기 = 누산기 + 메모리 값, 누산기 = 누산기 + 상수, 누산기 = 누산기 + 레지스터 값
  - 피연산자는 연산에 사용될 값을 의미
  - 피연산자로 주기억장치 주소, 상수, CPU 레지스터 번호 등이 가능
  - 위와 같이 다양한 형태의 피연산자를 어드레싱 모드(Addressing Mode)라 함
  - 피연산자 부분의 크기는 주기억장치의 크기에 영향을 받음

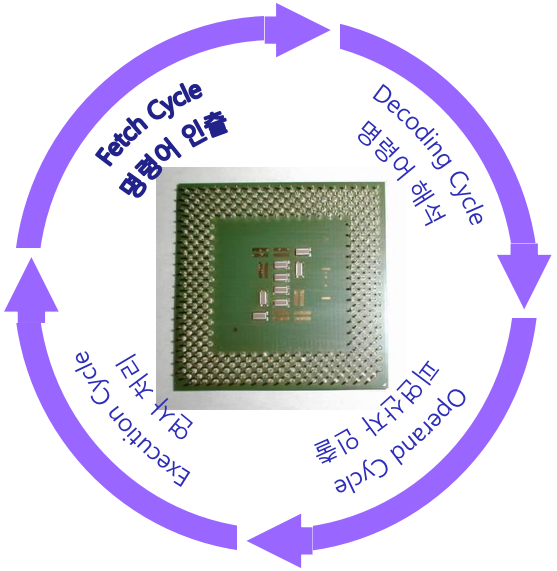


0. 교과 배경 지식...

0.1.4 기계 사이클(Machine Cycle)

- CPU의 일과
  - 주기억장치로부터 끊임없이 기계 명령어를 주소 번호대로(차례로) 인출하여 처리
  - 점프를 의미하는 기계 명령어를 만나면 명령어 인출 위치를 변경





0. 교과 배경 지식...

- 기계 사이클(Machine Cycle)
  - 인출 사이클 (Fetch Cycle)
    - 이번 차례 주소(PC가 가르키는 곳)의 주기억장치에서 명령어를 읽어 옴
    - 인출이 완료되면, PC는 다음 차례의 명령어 주소로 업데이트 됨
  - 해독 사이클 (Decoding Cycle)
    - 연산 코드를 분석하여 어떤 연산인지를 식별함
  - 피연산자 사이클 (Operand Cycle)
    - 피연산자 인출이 필요한 경우 지정된 곳에서 피연사자를 읽어옴
  - 실행 사이클 (Execution Cycle)
    - 누산기와 피연산자 사이에 연산을 적용하여 명령어를 실행함

❑ Fetch Cycle	• MAR ← PC • MBR ← (M) • PC ← PC + 2
❑ Decoding Cycle	• IR ← MBR(12~15)
❑ Operand Cycle	• MAR ← MBR(0~11) • MBR ← (M)
❑ Execution Cycle(AND operation)	• AC ← AC & MBR

0. 교과 배경 지식...

0.1.5 레지스터(Register)

- 레지스터란?
  - 값을 저장하는 곳 중의 하나
  - 메모리와 달리 불규칙적, 소량
  - 단순한 값의 저장 기능 외에, 저장 값에 따라 하드웨어 등의 상태에 영향을 미침
  - 레지스터는 CPU 내 번호, 주기억장치 주소의 일부 영역, 입출력 포트 번호 등에 의해 접근됨
- 레지스터의 유형
  - CPU 레지스터
    - CPU 내에 위치하여 CPU 제어, 연산(누산기), 값의 임시 저장 용도로 사용
  - 특수 기능 레지스터(SFR: Special Function Register)
    - 컴퓨터 전반에 걸친 상태를 설정하거나 현재 상태 표시
  - 입·출력 레지스터
    - 입 · 출력 장치와 연결되어 입력이나 출력할 때 데이터 전달 통로 역할

0. 교과 배경 지식...

0.1.6 인터럽트(Interrupt)

- 인터럽트란?
  - CPU에 전달되는 사건 신호(Event Signal)
  - 사건 신호에는 여러 가지가 있으며, 주로 각각의 전용 회선으로 전달됨

    ➔ Vectored Interrupt
  - 전용 회선이 없는 경우, 어떤 인터럽트가 발생했는지를 탐색해야 함

    ➔ Interrupt Polling(Polling Interrupt)

0. 교과 배경 지식...

- 인터럽트 서비스 루틴(ISR: Interrupt Service Routine)
  - 인터럽트 핸들러(Interrupt Handler)라고도 함
  - 발생한 인터럽트에 대한 CPU의 반응(대응) 처리를 위한 프로그램 코드
  - 개발자가 인터럽트마다 고유한 ISR을 작성하여 배치

- 인터럽트 벡터(Interrupt Vector)
  - 인터럽트별 ISR의 위치 정보를 보관하는 장소(물론 주기억장치 내에 있음)
  - 기계에 따라 인터럽트 벡터의 위치가 고정되기도 하고, 설정 가능하기도 함
  - 인터럽트 벡터의 크기는 기계에서 허용되는 최대 인터럽트 수에 의해 결정됨
    - Z80 : 128개 (2 바이트 주소 공간→ 128 * 2 = 256 바이트 차지)
    - Pentium : 256개(4 바이트 주소 공간→ 256 * 4 = 1024 바이트 차지)

      ➔ 인터럽트 벡터 레지스터에 벡터의 위치를 설정함
    - Atmega128: 58개(2 바이트 주소 공간) → 58 * 2 = 116 바이트 차지)

      ➔ 주기억장치의 0번지에 고정됨

0. 교과 배경 지식...

- 2.2.3 인터럽트 우선 순위(Interrupt Priority)
  - 여러 개의 인터럽트가 동시에 발생한 경우 대응 처리 순서를 결정함
  - 기 발생한 인터럽트에 대한 대응 처리 도중, 다른 인터럽트가 발생했을 때, 그 인터럽트를 보류 시킬 것인지 아니면 지금 즉시 처리할 것인지를 결정함

    ➔ 새로 발생한 인터럽트의 우선 순위가 더 높으면 진행 중이던 대응 처리를 잠시 유보하고 새로운 인터럽트 처리를 먼저 처리한 후 재개함

- 2.2.4 인터럽트 사이클(Interrupt Cycle)
  - CPU가 인터럽트 발생 여부를 체크(조사)하는 시기를 말함
  - [그림 2-4]의 4 단계 기계 사이클을 마치고 인터럽트 발생 여부를 조사함
    - 즉, 하나의 기계 명령어에 대한 처리를 마칠 때마다 조사함
    - 결국, 기계 사이클은 인터럽트 사이클을 포함하여 총 5 단계로 이루어짐

0. 교과 배경 지식...

- 2.2.5 인터럽트 유형
  - 디바이스 인터럽트(Device Interrupt)
    - 입·출력 장치 등 CPU 외부 주변 기기에서 발생하는 인터럽트

      → 하드웨어 인터럽트(Hardware Interrupt)라고도 함
  - 오류 인터럽트(Error Interrupt)
    - CPU가 기계 명령어를 처리하는 도중에 발생하는 인터럽트
    - 잘못된 기계 명령어를 만나거나 0으로 나누는 등의 연산 불가 상황에서 발생

      → 예상하지 않는 오류 발생이란 뜻에서 예외(Exception)라고 함
  - 소프트웨어 인터럽트(Software Interrupt)
    - 인터럽트를 발생시키는 기계 명령어 즉, 프로그램에 의해 발생된 인터럽트
    - 인터럽트 대응 처리를 테스트하거나, 시스템 콜을 위해 사용자 프로그램에서 운영체제로 진입하기 위해 사용

      → 프로그램의 흐름을 인위적으로 잠시 특별한 부분으로 빠져들게 하므로 트랩(Trap)이라고도 함

0. 교과 배경 지식...

0.1.7 입•출력 장치

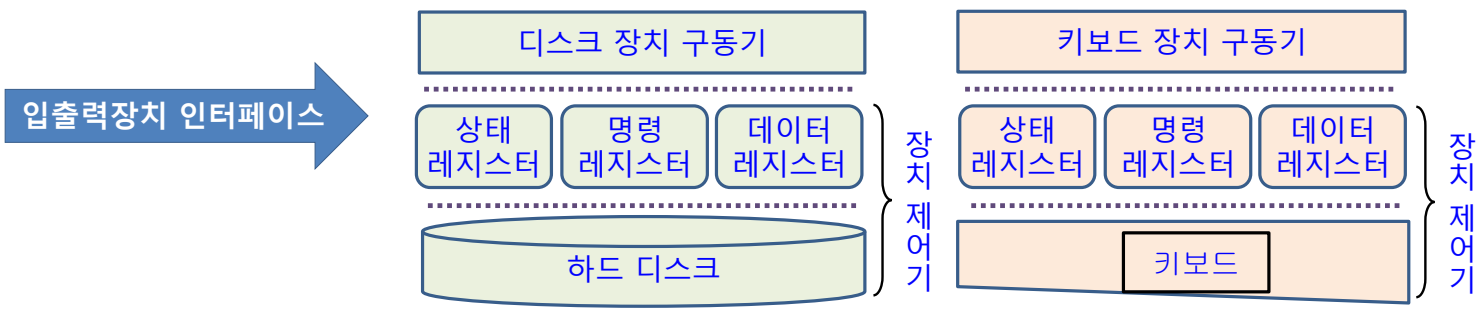
- 입력(Input)
  - CPU, DMA 등 처리기가 주변장치의 데이터 레지스터(Data Register)로부터 데이터 값을 읽어 주기억 장치로 복사하는 작업
  - 데이터 레지스터로부터 값을 읽을 때에는 상태 레지스터(Status Register)의 데이터 준비 상태를 확인해야 함

    → 준비되지 않은 상태에서 읽을 경우 엉터리 값을 읽게 됨
- 출력(Output)
  - 입력과 반대로 주기억 장치의 데이터 값을 읽어 주변장치의 데이터 레지스터로 복사하는 작업
  - 데이터 레지스터에 값을 쓰기할 때에는, 상태 레지스터에서 이전 데이터의 출력이 완료되었는지 확인해야 함

    → 이전 데이터의 출력 완료 전 또 다른 데이터를 쓰기하면 출력 데이터의 정확성이 보장되지 않음

0. 교과 배경 지식...

- 입출력 장치 구조
  - 입·출력 장치 제어기(IO Device Controller) : 입·출력 장치 인터페이스
    - 상태 레지스터(Status Register)
      - 데이터가 입력되었는가 혹은 데이터 출력이 완료되었는가 등 장치의 상태를 표시하는 곳
    - 명령 레지스터(Command Register)
      - '입력' 혹은 '출력' 하라는 등의 명령을 보내는(쓰는) 곳
      - '입력' 명령에 따라 입력 ➔ 동기적 입력(디스크 등)
      - '입력' 명령없이 수시로 입력 ➔ 비동기적 입력(키보드 등)
      - 출력은 언제나 동기적
    - 데이터 레지스터(Data Register)
      - 입력된 데이터나 출력될 데이터를 임시 보관하고 있는 곳
  - 입출력 장치 구동기(IO Device Driver)
    - 장치 제어기 인터페이스를 이용하여 입·출력을 처리하는 소프트웨어



0. 교과 배경 지식...

- 2.4.2 입·출력 장치의 식별
  - 메모리 대응 입·출력(Memory Mapped IO)
    - 입·출력 장치의 레지스터들의 위치를 주기억장치의 주소영역 일부에 대응
    - 주기억 장치(변수, 주소)에서 읽고 쓰기하는 동일한 방법으로 입출력이 이루어짐
    - 즉, 메모리 접근 명령어와 동일한 기계 명령어(load, save 등)를 사용하여 입·출력을 진행함

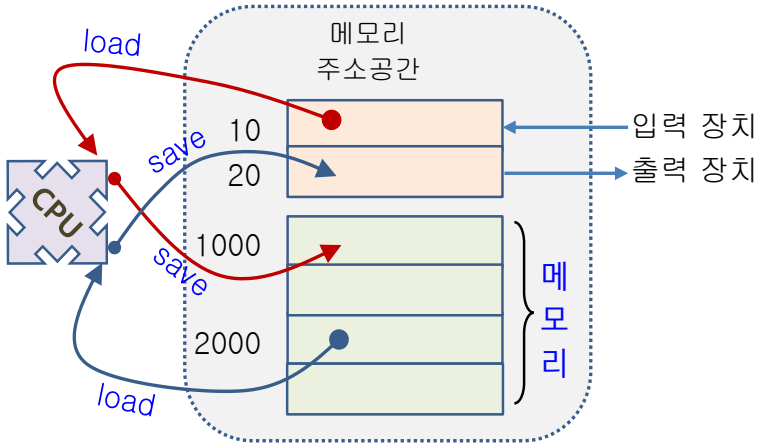
      → 개발자 입장에서 편리하나, 메모리 공간의 일부는 사용할 수 없음

입력 예) load r1, 10 ⇒ 메모리 10번지에 대응된 입력 레지스터에서 CPU의 r1 레지스터로 읽음(입력)

save r1, 1000 ⇒ 입력된 데이터를 메모리 1000번지에 저장

출력 예) load r1, 2000 ⇒ 메모리 2000번지의 데이터를 CPU의 r1 레지스터로 적재

save r1, 20 ⇒ 적재된 데이터를 메모리 20번지에 대응된 출력 레지스터에 기록(출력)



0. 교과 배경 지식...

- 격리된 입·출력(Isolated IO)
  - 주기억 장치의 주소 영역과 별개로 입·출력 장치 주소 영역(포트, Port)이 따로 있음
  - 즉, 메모리 10 번지와 입·출력 포트 10의 숫자는 동일하지만 지시하는 위치는 전혀 다름
  - 따라서, 메모리 접근 기계어 외에 입·출력 포트 접근을 위한 전용 기계 명령어(in, out 등)가 필요함

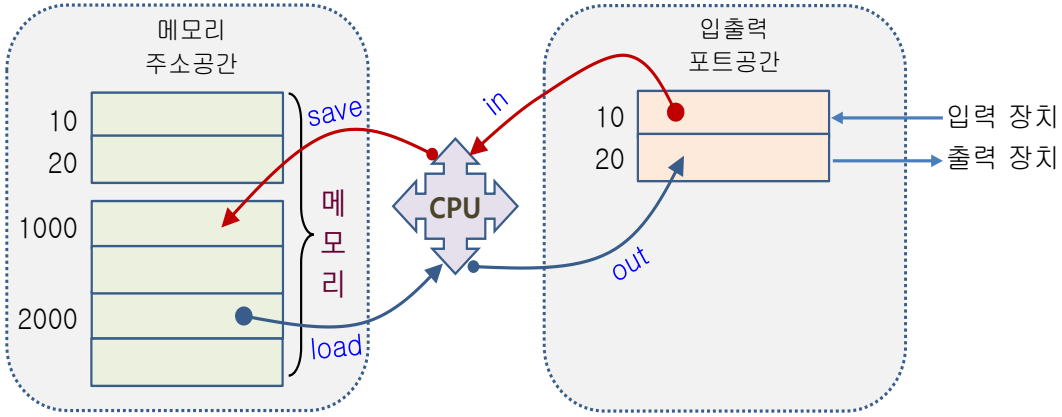
    → 메모리 공간을 모두 사용할 수 있으나, 시스템 설계가 다소 복잡해 짐

입력 예) in r1, 10 ⇒ 입력 포트 10번지의 입력 레지스터에서 CPU의 r1 레지스터로 읽음(입력)

save r1, 1000 ⇒ 입력된 데이터를 메모리 1000번지에 저장

출력 예) load r1, 2000 ⇒ 메모리 2000번지의 데이터를 CPU의 r1 레지스터로 적재

out r1, 20 ⇒ 적재된 데이터를 메모리 20번지의 출력 레지스터에 기록(출력)



0. 교과 배경 지식...

0.1.8 컴퓨터 저장 장치

- 컴퓨터에 사용되는 저장장치는 [그림 2-10]과 같이 가격, 성능, 용량에 따라 계층을 이루고 있다.

⊙ 레지스터
- CPU 내에 존재하는 연산 및 임시 저장 용도의 저장 공간

⊙ 캐시
- CPU와 주기억장치 사이의 속도차 극복을 위한 소량, 고속, 고가 메모리
- 프로그램의 지역성 특성에 의거 90% 이상의 적중률

⊙ 전자 디스크
- 플래시 메모리 기술에 기반한 SSD(Solid State Disk)