Chapter. 6

Embedded System I

Embedded Processor

국립 한밭대학교 HANBAT NATIONAL UNIVERSITY

Professor.

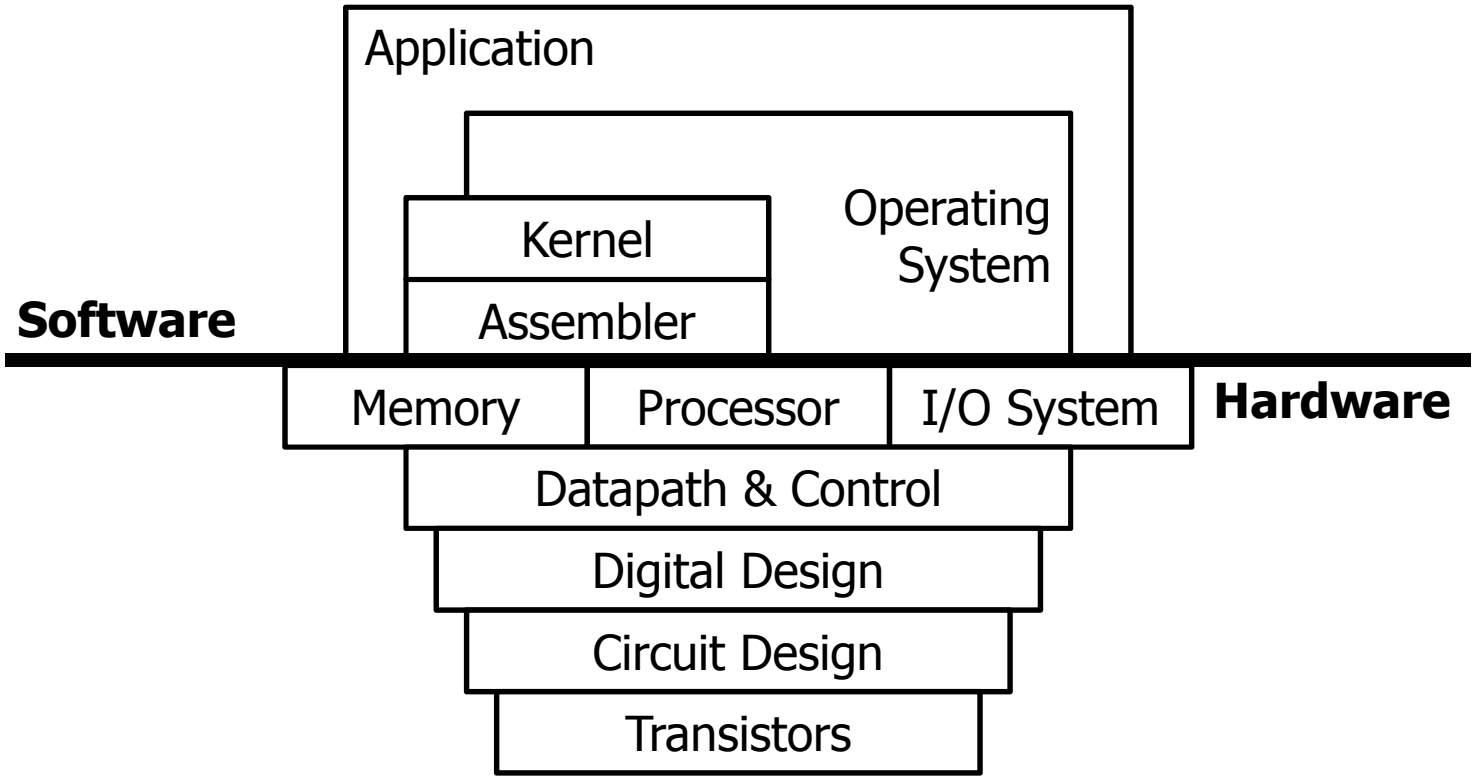
Jaeheung, Lee

목차

- Embedded System 구조
- Embedded Processor
- Embedded System

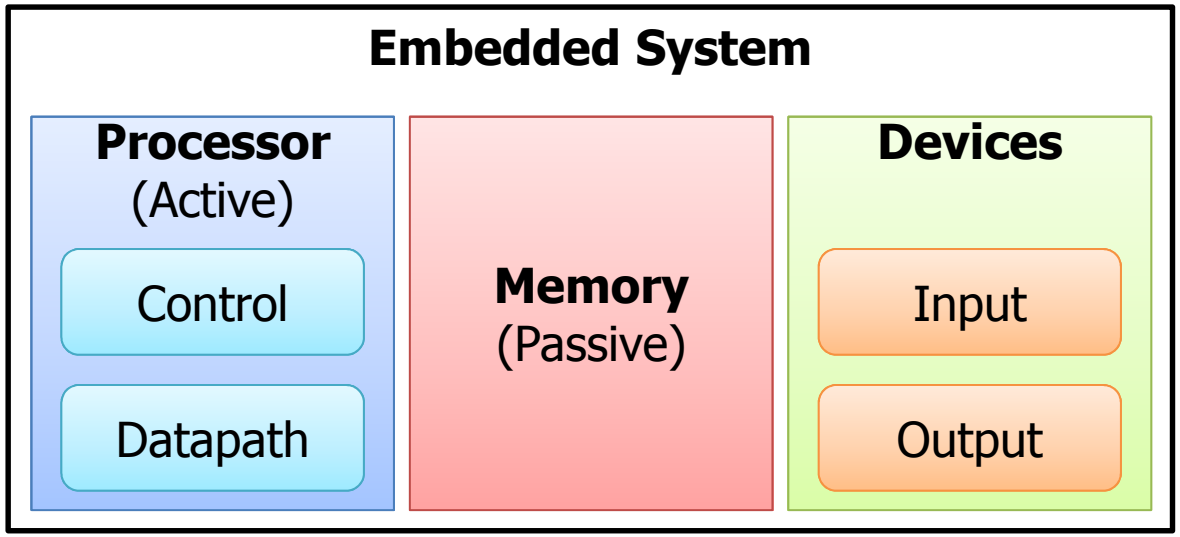
Embedded System 구조

- System Architecture



Embedded System 구조

- Embedded System 구성
  - Embedded H/W
    - Processor, Controller
    - Memory, Bus, I/O Interface, Network Interface, …

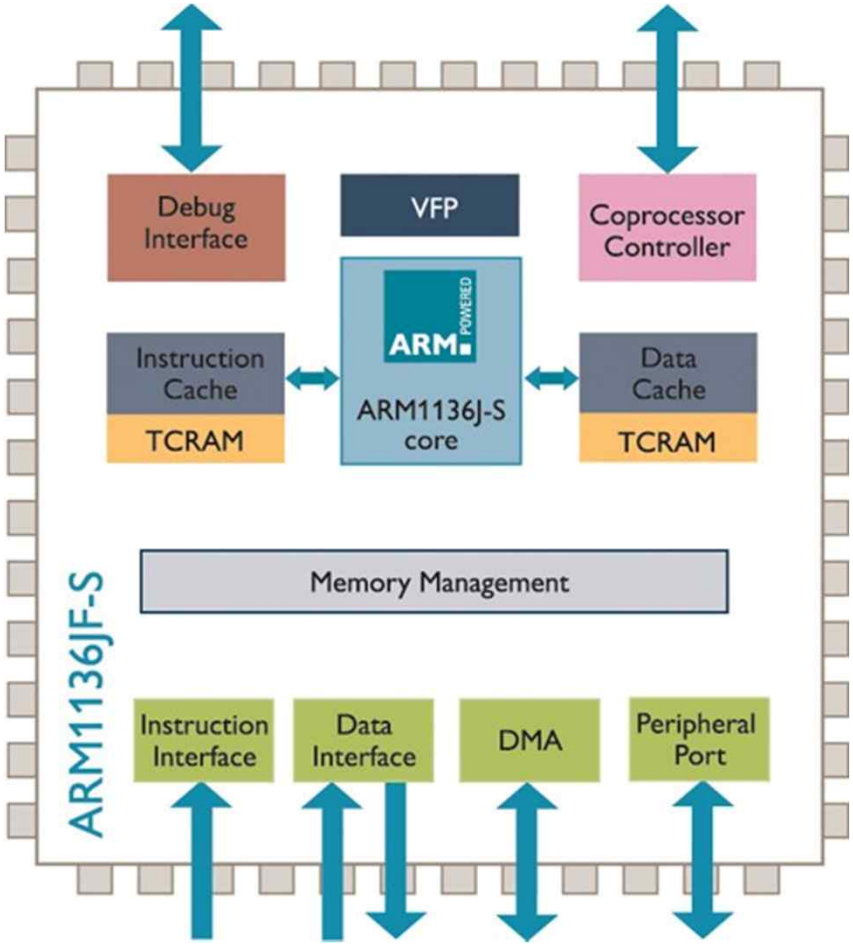


Embedded Processor

- Embedded Processor 역할
  - Computation Task를 담당
  - 다양한 주변 인터페이스를 포함하는 SoC 형태로 발전
  - 처리속도, 전력소비, 가격 뿐만 아니라 개발 환경과의 연관관계가 중요
  - 제어 장치(Control Unit)와 연산부(Data-Path)로 구성
  - 프로세서 선택의 다양성
    - **ARM**, MIPS, i386, Alpha, Sparc, m68K

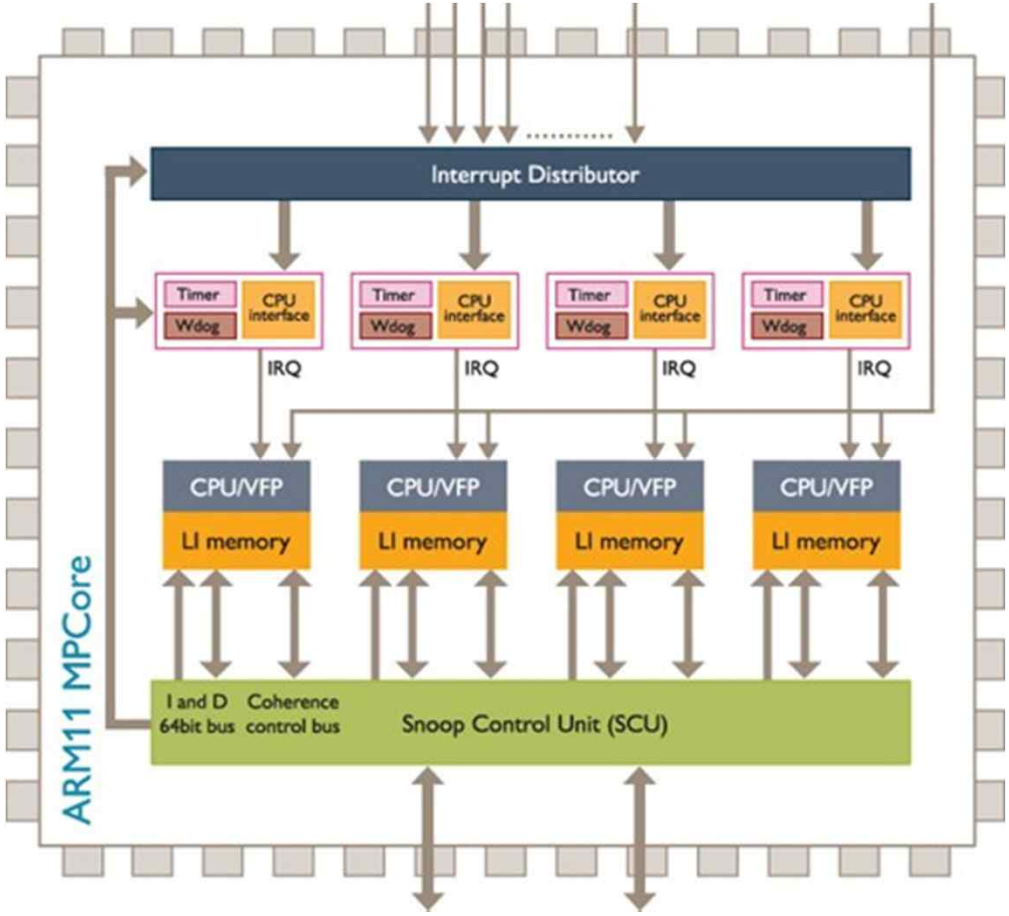
Embedded Processor

- Embedded Processor 구조 - ARM1136J-S (Single-Core)



Embedded Processor

- Embedded Processor 구조 - ARM11MP (Multi-Core)

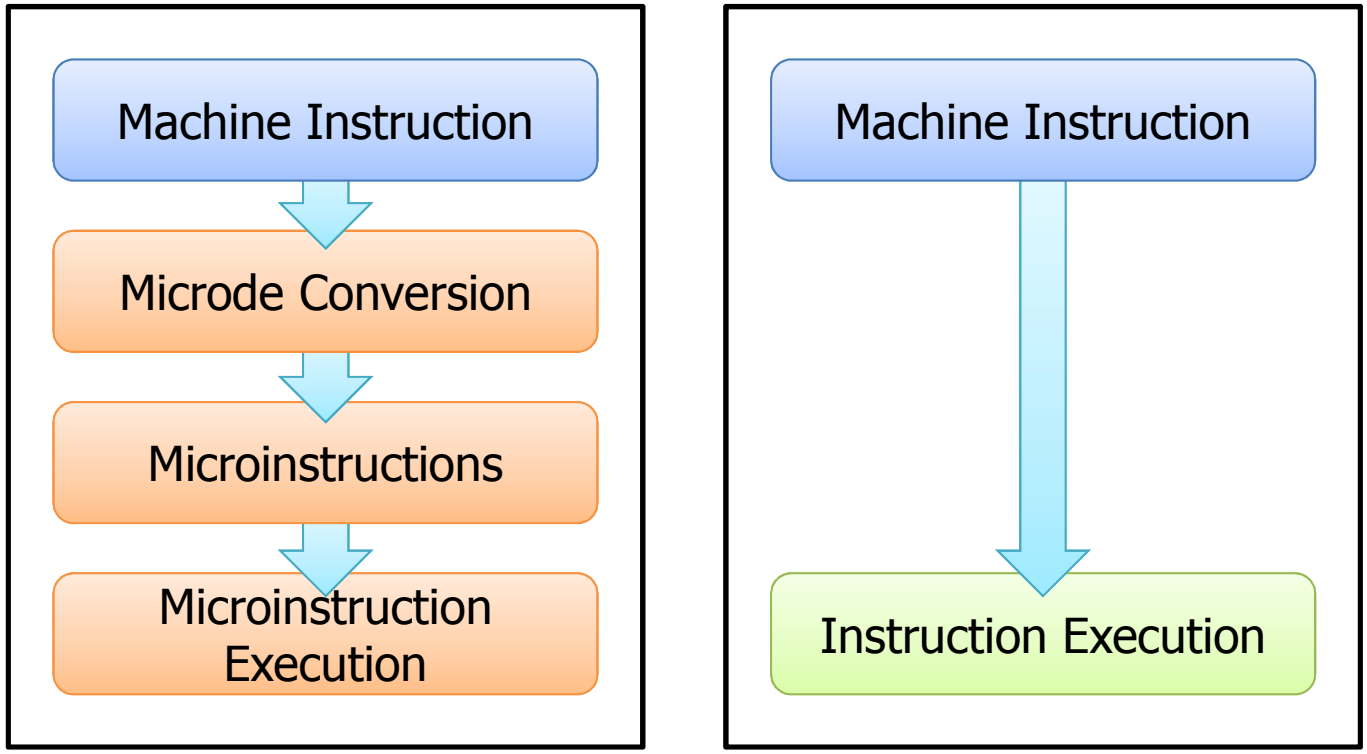


Embedded Processor

- CISC and RISC
  - Complex Instruction Set Computer (CISC)
    - 관련된 연산을 수행하는 수많은 명령을 보유
    - 각 명령마다 많은 클럭 사이클이 소요
    - 많은 실리콘 영역을 차지함 (높은 생산가격)
  - Reduced Instruction Set Computer (RISC)
    - CISC에 비해 간단한 구조
    - 클럭 사이클당 한 개의 명령이 실행 (매우 빠른 속도)
    - RISC Core들의 소형화

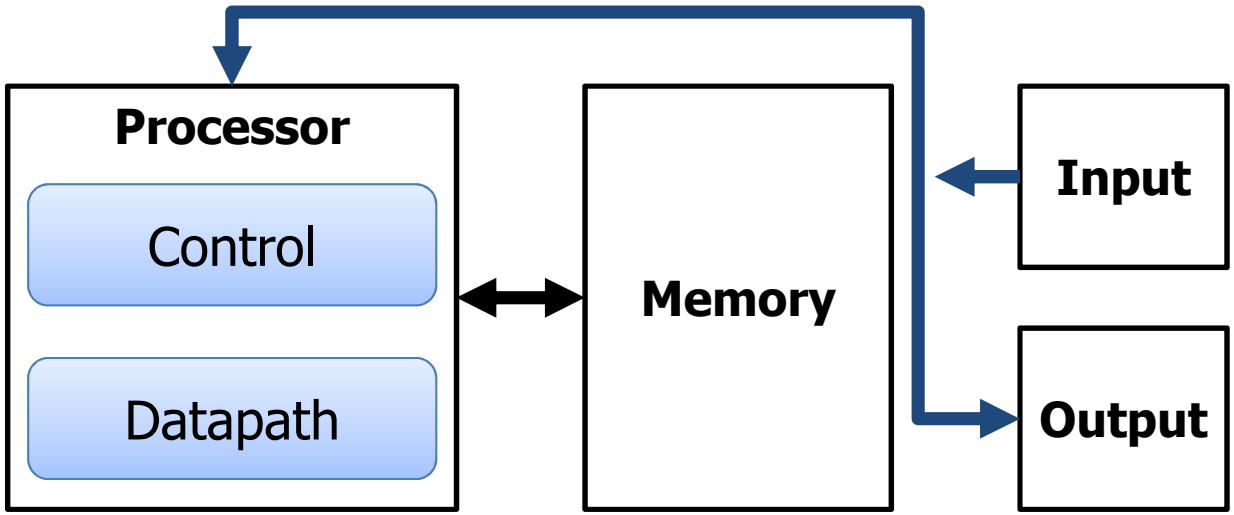
Embedded Processor

- CISC and RISC



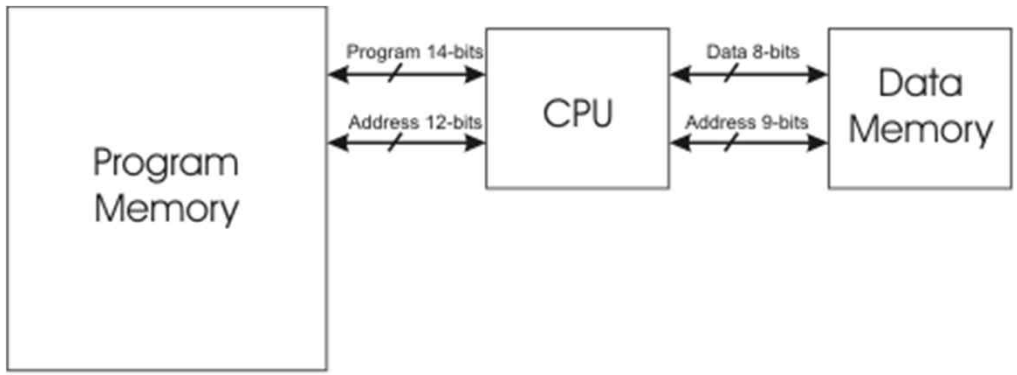
Embedded System

- BUS
  - 공유되는 통신 연결통로
  - 여러 Subsystem들을 연결하는데 쓰이는 하나의 Write Set
  - Data Bus, Address Bus, Control Bus



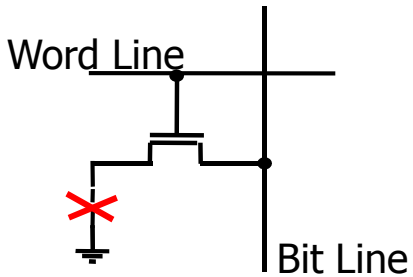
Embedded System

- Havard Architecture
  - 명령용 버스와 데이터용 버스로 물리적으로 분할한 Architecture
    - 동시에 두 개의 메모리 패치가 가능
  - Von-Neumann Architecture에서 Memory 접근의 병목현상을 개선하기 위해 제시된 Architecture
    - 병목현상 : CPU와 기억장치 사이의 단 하나의 통로에 정보가 집중됨으로써 야기되는 작업 지연 현상

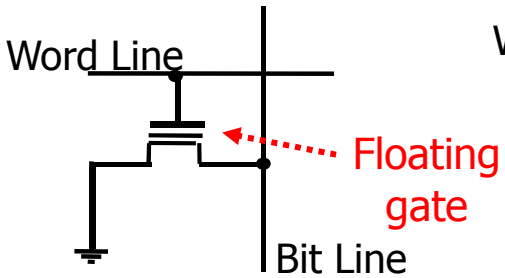


Embedded System

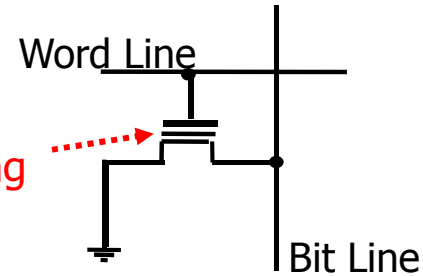
- Memory – ROM
  - Read-Only Memory (ROM)
    - Non-Volatile Storage
    - Type : ROM, PROM, EPROM, EEPROM
    - OT-PROM (One Time Programmable)
      - Mask ROM
      - Fuse ROM
    - PROM (Programmable)
      - EPROM
      - EEPORM



Mask ROM
Fuse ROM



EPROM



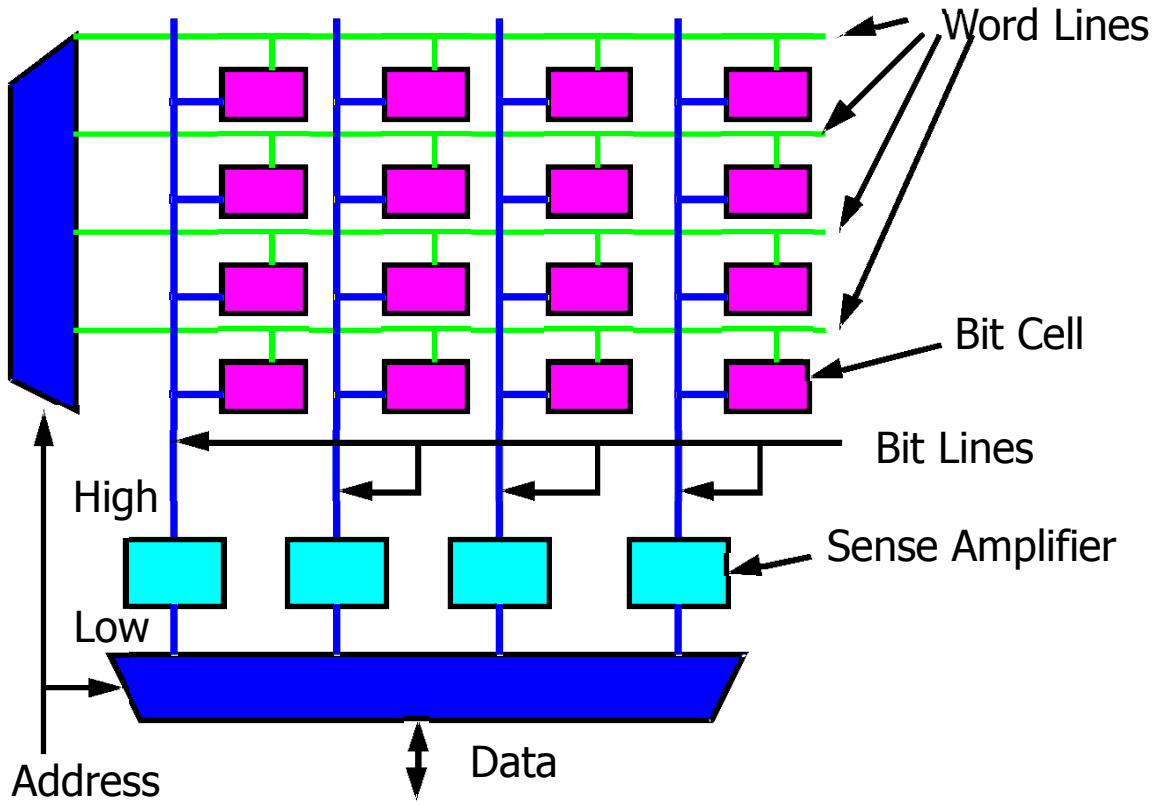
EEPROM
Flash Memory

Embedded System

- Memory – RAM
  - Random Access Memory (RAM)
    - Volatile Storage
    - Type : Static RAM (SRAM), Dynamic RAM (DRAM)
    - Static RAM
      - 속도가 빠름
      - 저밀도 (4-6 Transistors/bit)
      - Power가 공급되는 동안 데이터를 안정적으로 저장/유지
    - Dynamic RAM
      - 속도가 느림
      - 고밀도 (1 Transistor/bit)
      - 불안정 – Refresh 필요

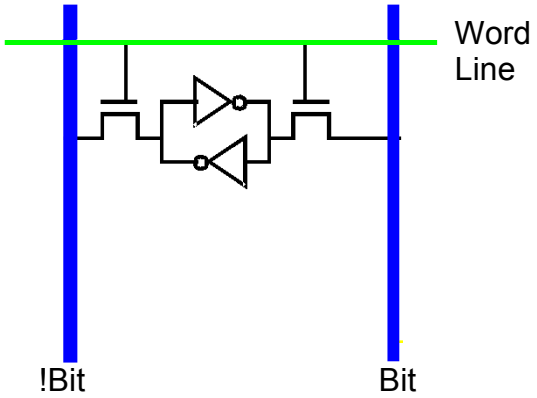
Embedded System

- Memory – RAM
  - Random Access Memory (RAM)의 기본 구조

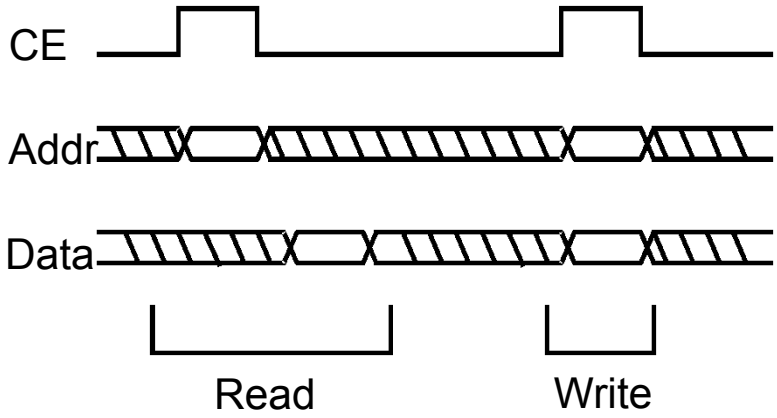


Embedded System

- Memory – RAM
  - Static RAM 구조 및 Access



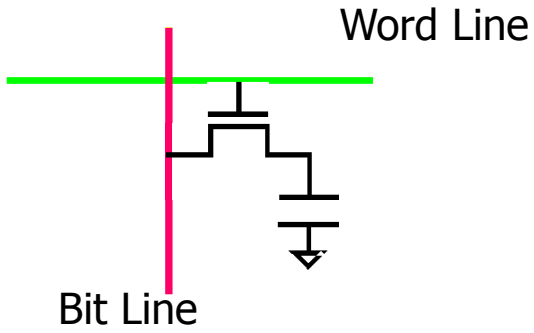
Read: word line을 이용, sense bit line들의 value의 값을 얻어냄
Write: word line을 이용, bit line에 새로운 값을 write



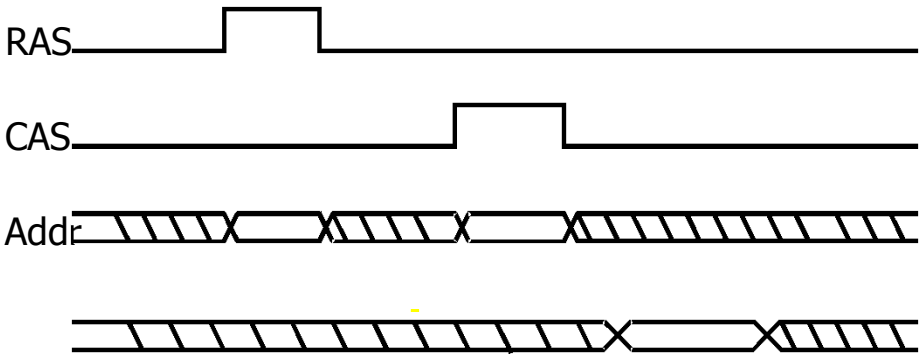
Accessing a Static RAM
Note: CE(Chip Enable)시그널은 빈번하게 active-low됨. SRAM은 일반적으로 write enable signal을 가지고 있음

Embedded System

- Memory – RAM
  - Dynamic RAM 구조 및 Access



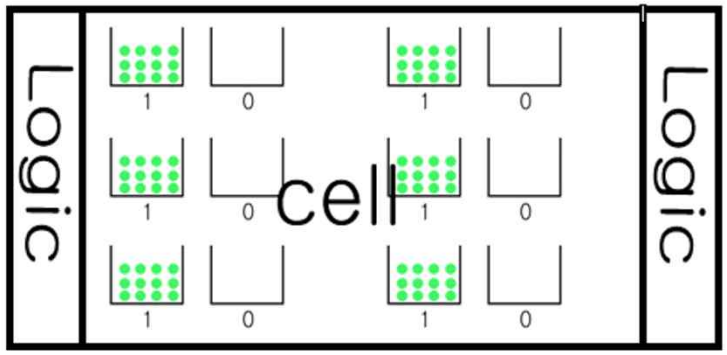
Read: word line을 이용, bit line에 있는 값을 읽어옴.기존에 저장되어 있던 저장된 값을 지움(refresh)
Write: word line을 이용, bit line에 새로운 값을 write함



Dynamic RAM Timing (Read)
Control signal은 빈번하게 active-low됨

Embedded System

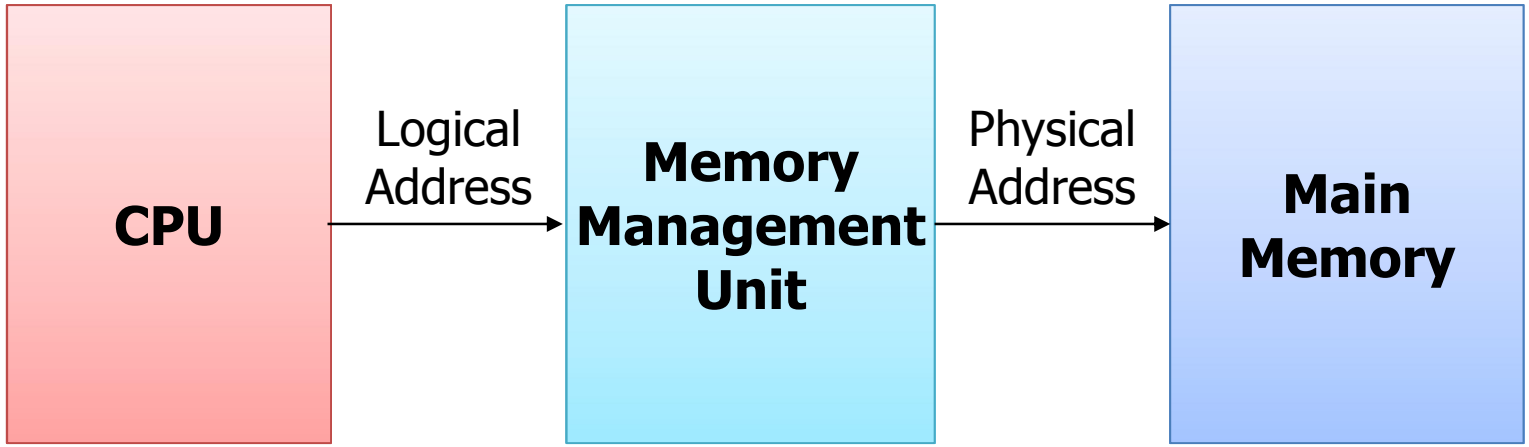
- Memory – Other Type
  - Nand Flash Memory
    - Flash Memory의 한 종류로 전원이 없는 상태에서도 데이터를 계속 저장할 수 있으며, 데이터를 자유롭게 저장/삭제할 수 있음
    - 저장단위인 셀을 수직으로 배열해 좁은 면적에 많은 셀을 만들 수 있도록 설계되어있어 대용량이 가능
    - 디지털 카메라, 휴대폰, 컴퓨터, USB 등에 널리 사용됨



NAND Flash의 구조

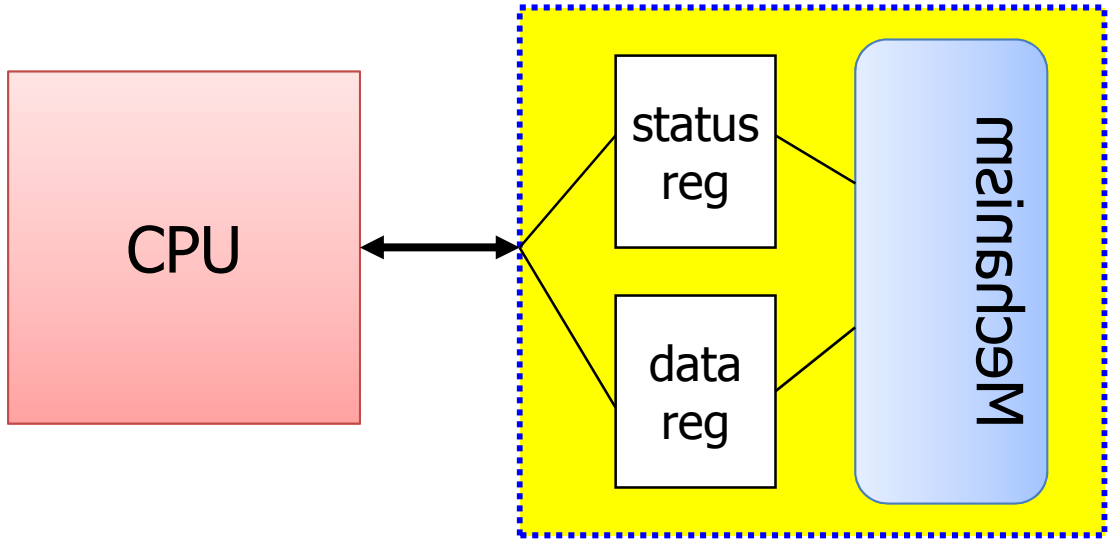
Embedded System

- Memory Management Unit (MMU)
  - 주소를 변환
    - Physical Address -> Logical Address
  - 프로그램이나 컴퓨터 시스템에 대해서 권한이 없는 접근을 차단하는 Protection Checks



Embedded System

- I/O Devices
  - 전형적으로 CPU와 Digital Interface로 연결



Embedded System

- I/O Addressing
  - Microprocessor는 I/O 핀들을 이용함으로써 다른 Device들과 통신함
  - 포트기반 I/O (병렬 I/O)
    - 프로세서는 하나 또는 그 이상의 N-bit Port들을 가짐
    - 프로세서의 소프트웨어는 레지스터처럼 Port를 Read/Write 함
  - Bus기반 I/O
    - 프로세서는 하나의 버스를 생성하는 Address, Data and Control Port들을 가짐
    - 통신 프로토콜은 프로세서 안에 내장됨
    - 각각의 하나의 Instruction은 버스에서 Read 또는 Write 프로토콜을 수행함

Embedded System

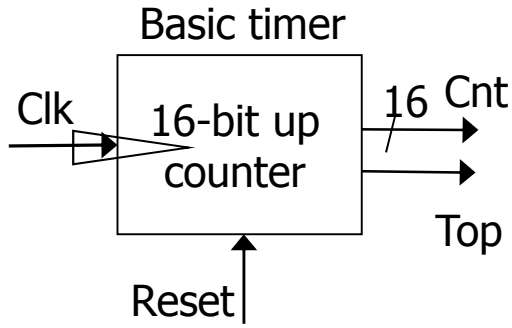
- Bus 기반 I/O
  - 프로세서는 동일한 버스를 사용해서 메모리나 주변장치와 통신
  - Memory-mapped I/O
    - 주변장치 레지스터는 같은 메모리 주소 공간을 나눠 점유함
    - 예) Bus는 16-bit 어드레스(64K)를 가짐
      - 그 중 하위 32K 어드레스는 메모리로 할당
      - 그 중 상위 32K 어드레스는 주변장치에 할당
  - Standard I/O (I/O-mapped I/O)
    - 버스의 추가적인 핀으로(M/IO) 메모리 또는 주변장치로의 접근을 판별한다
    - 예) Bus는 16-bit 어드레스를 가짐
      - 모든 64K 어드레스는 M/IO 이 0이 될 때 메모리에 할당
      - 모든 64K 어드레스는 M/IO 이 1이 될 때 메모리에 할당

Embedded System

- Memory-mapped vs. Standard I/O
  - Memory-mapped I/O
    - 다른 특별한 명령이 요구되지 않음
      - MOV 와 ADD 와 같이 메모리를 필요로 하는 어셈블리 Instruction은 주변장치와도 잘 동작
  - Standard I/O
    - 주변장치로의 메모리 어드레스의 손실이 없음
    - 주변장치들에 있는 더욱 단순한 어드레스 Decoding Logic
      - 주변장치의 수가 어드레스 공간보다 적은 경우 가중치가 높은 고차 어드레스 비트들은 무시
        - Smaller and/or Faster 비교기
      - Standard I/O는 주변기기 레지스터와 메모리 레지스터 사이에서 데이터를 move하기 위해 특정instruction(예를 들어 IN,OUT같은) 들을 필요

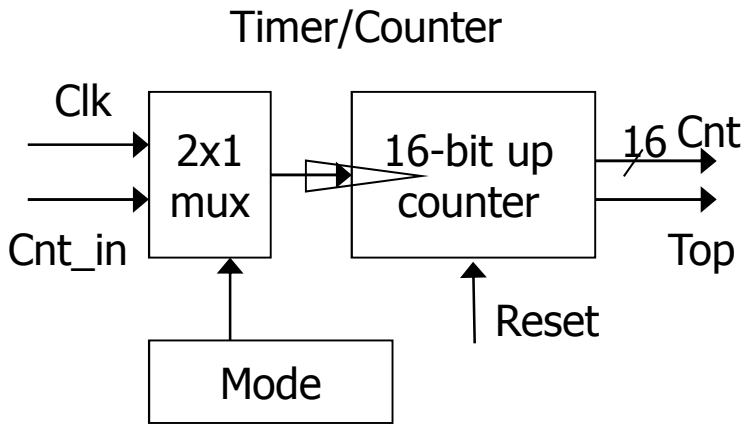
Embedded System

- Timer
  - 시간 간격(time interval) 측정
  - 시간과 관계된 출력 이벤트를 생성하기 위해
    - 예) 녹색 신호등을 10초간 유지
  - 입력 이벤트를 측정하기 위해
    - 예) 차의 속력을 측정
  - 클럭 펄스의 counting에 기반
    - 1. 클럭 주기를 10 ns 로 조정
    - 2. 20,000 Clk 펄스를 세면
    - 3. 200 microseconds가 지나게 됨
    - 16-bit는 최대 65,535*10 ns = 655.35ms를 셀 것. (Resolution = 10 ns)
    - 최대카운트에 도달하면 처음으로 되돌아감



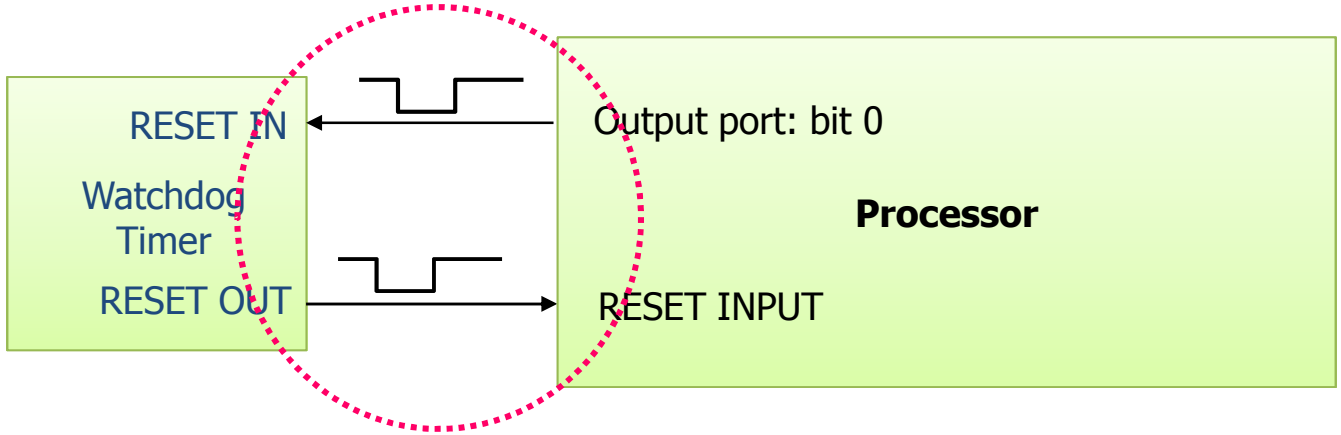
Embedded System

- Counter
  - 카운터와 유사하나, 클럭 펄스의 수를 세는 것이 아니라 일반 입력 신호로부터의 펄스 수를 카운트
    - 예) 센서를 지나가는 차의 개수를 셈
    - 타이머 또는 카운터로 자유롭게 쓸 수 있음



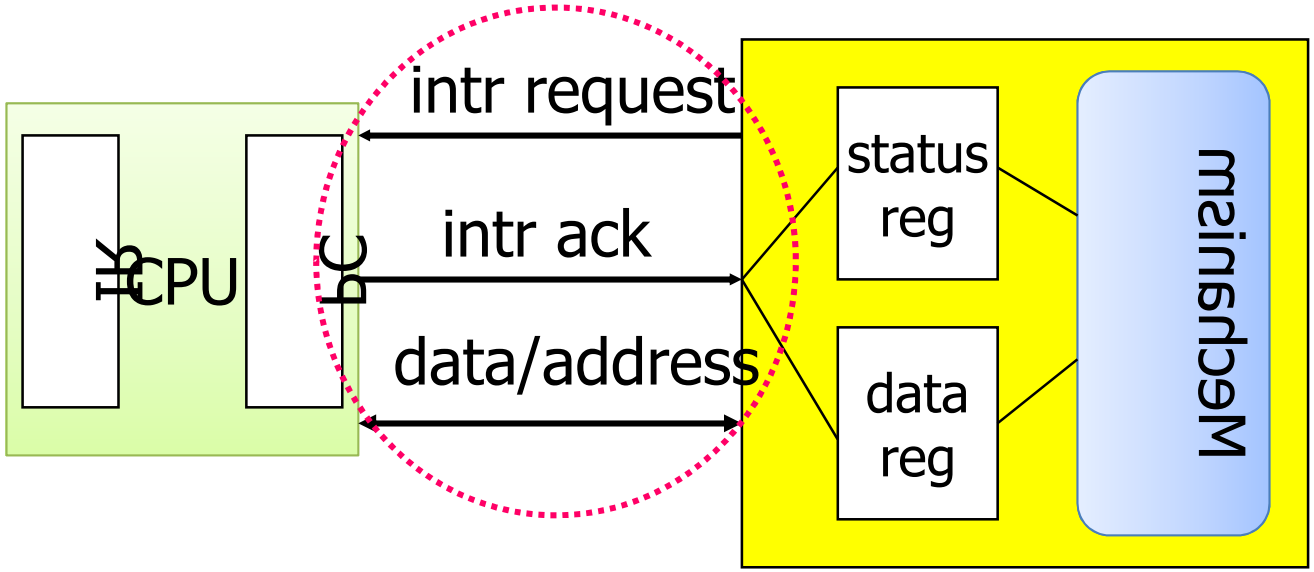
Embedded System

- Watchdog Timer
  - 대부분의 산업적이거나 특수한 기능에 쓰이는 임베디드 시스템은 오류가 나면 안 되는데, 그렇다면 우리는 어떻게 glitch가 Instruction의 흐름을 파괴하지 못하게 보장할 수 있겠는가?
  - Watchdog Timer : 시스템의 동작을 Monitoring하여, 다양한 조건 발생 시에 RESET Signal 발생
    - Power supply voltage가 일정 범위를 벗어 나는 경우
    - 컴퓨터가 지정된 time interval 에서 Reset 펄스를 내지 못하는 경우



Embedded System

- Interrupt Interface
  - Embedded System의 실시간성 요구에 필수적인 요소



Embedded System

- Interrupt Interface – cont'd
  - ISR(interrupt service routine)의 주소?
    - Fixed Interrupt
    - 어드레스는 마이크로 프로세서에 내장되어 있어서 바뀔 수 없음
    - Either ISR stored at address or a jump to actual ISR stored if not enough bytes available
    - 가용한 바이트수가 충분하지 않은 경우에는, ISR이 어드레스에 저장되거나 저장된 실제 ISR로 점프함
  - Vectored Interrupt
    - 주변장치가 주소를 제공
    - 마이크로 프로세서가 시스템 버스에 의해 연결된 여러 주변 장치들을 가지고 있는 것이 일반 적인 경우

Embedded System

- Additional Interrupt Issues
  - Maskable vs. Non-Maskable 인터럽트
  - Maskable: 프로그래머는 프로세서에게 인터럽트를 무시하도록 하는 비트를 set할 수 있음
  - time-critical 한 코드에 사용하여 인터럽트를 무시하도록 함
  - Non-Maskable:마스크 될 수 없는, 구별된 인터럽트 핀
    - 전형적으로 위급한 상황에 쓰이는데, 갑작스런 정전 시, 즉시 비-휘발성 메모리에 데이터를 백업하는 것이 요구되는 경우가 이에 속함
  - Jump to ISR
    - 몇몇 마이크로 프로세서는 서브루틴의 호출과 jump를 같은 것으로 다룸
    - 완벽하게 저장된 상태 (PC, registers) – 수백 개의 사이클이 소요됨
    - 다른 것들은 PC처럼 부분적인 상태만을 저장
    - 그래서, ISR 은 레지스터를 수정하지 말아야 한다. 그것이 아니라면 레지스터를 처음에 저장
    - Assembly-language 프로그래머는 어떤 레지스터가 저장되었는지를 꼭 확인

Embedded System

- Direct Memory Access (DMA)
  - Buffering
    - 프로세싱 전에 일시적으로 데이터를 메모리에 저장
    - 데이터는 일반적으로 버퍼가 있는 주변 장치에 축적
  - 마이크로프로세서는 ISR을 가지고 그것을 해결할 수 있음
    - 하지만 비효율적으로 마이크로프로세서는 상태를 저장하고 또 재저장해야 하는 점과 다른 보통의 프로그램이 그것을 기다려야만 하는 점이 있음
  - DMA는 좀 더 효율적
    - 단일 목적을 가진 프로세서를 구분함
    - 마이크로 프로세서는 시스템 버스의 컨트롤을 DMA컨트롤러에게 양도
    - 마이크로 프로세서는 보통 프로그램의 안정적인 실행을 보장함
      - ISR 호출 때문에 상태를 저장하고 또 저장하는 비효율성의 해소
      - 보통 프로그램은 시스템 버스를 요구하는 것이 아니라면 기다리지 않아도 됨