14주 1강

병렬 처리의 개념과 병렬 컴퓨터의 분류

병렬처리의 개념

- 병렬처리(parallel processing)
  - 여러 개의 프로세서를 장착하고 하나의 프로그램을 서로 다른 태스크(task)로 동시에 처리할 수 있는 방식
  - 처리 부하를 분담하고, 처리 속도를 향상시킬 수 있다.
- 등장배경
  - 단일 프로세서의 기술 향상과 구조적 개선의 한계성에 직면
  - VLSI 기술에 의해 작고 저렴한 고속의 프로세서들을 하나의 시스템으로 구성 가능, 병렬 컴퓨터 구조의 구현을 가능하게 함.
  - 소프트웨어적으로 프로그램을 분할해서 처리할 수 있는 병렬 프로그램 언어와 컴파일러가 개발
  - 상호 공유자원에 대한 경합을 줄이고 이용률을 극대화할 수 있는 운영체제가 개발
- 복잡한 처리가 추가되기 때문에 반드시 처리장치의 개 수만큼의 속도가 향상되지는 않는다.

병렬 처리에서 고려할 문제- 분할과 스케줄링

- 분할 문제(Partition)
  - 그레인(grain) : 분할된 프로그램 부분
    - 크기가 매우 작으면 많은 병렬성을 얻지만 동기화와 스케줄링에 많은 과부하 발생
    - 크기가 크면 동기화와 스케줄링의 과부하가 줄어들지만 병렬성 떨어짐
- 스케줄링(Scheduling)
  - 분할된 태스크를 실행하기 위해서, 태스크들을 각 프로세서에 배정하는 것
  - 정적 스케줄링
    - 정해진 스케줄은 실행되는 동안 변하지 않는다.
    - 스케줄링 비용이 컴파일 시간에만 들고 실행 시에는 소요되지 않는 장점
    - 태스크의 실행 시간과 통신 비용 등을 정확히 예측하지 못하면 비효율적
  - 동적 스케줄링
    - 프로그램이 실행될 때 각 태스크를 프로세서에 할당하는 방법이다.
    - 장점은 프로세서의 이용률을 높일 수 있다.
    - 단점은 프로그램 실행 시에 스케줄링을 하므로 실행에 많은 부담을 준다.

병렬 처리에서 고려해야 할 문제- 동기화

- 동기화(Synchronization)
  - 각 프로세서에 공유된 데이터가 임의의 한 프로세서에 의해 변경되지 않도록 하고 정확한 값을 유지할 수 있도록 하기 위한 작업

  - 버스 잠금(Bus-locking) 방식
    - 한 프로세서가 버스를 이용하여 공유기억장치에 액세스할 때, 즉시 버스를 독점한 후 해당 영역을 처리하게 하는 방식이다.
    - 버스가 독점되므로 다른 프로세스들은 버스에 연결된 다른 자원을 사용할 수 없다.
    - 프로세스의 실행이 종료될 때까지 버스를 독점하여, 잘못된 데이터 정보가 공유하지 않도록 하는 방식이다.

  - 상태 표시(Flag) 방법
    - 데이터를 저장하는 자원 각각에 1비트의 상태 레지스터를 두어서 자원의 상태를 표시하도록 한 방식이다.
    - 대기 중인 프로세서는 상태 레지스터를 검사하여 사용 가능으로 되었을 때에만 액세스할 수 있다.

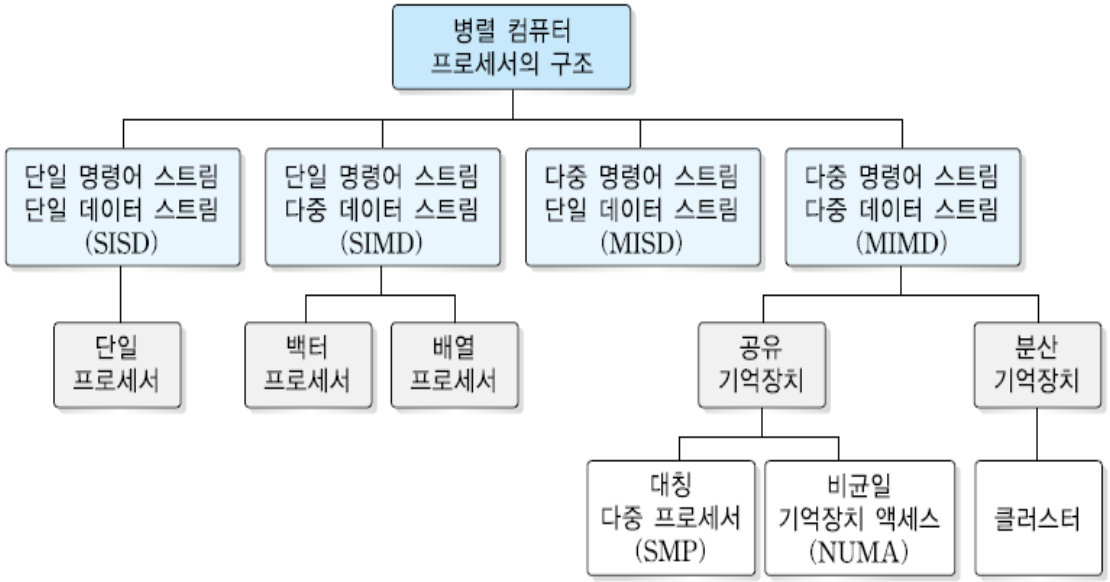
병렬 컴퓨터 분류

- 제어장치(Control Unit)와 처리장치(Processing Unit)
  - 제어장치는 명령어를 인출하고 해독하여서 제어신호를 만드는 장치
  - 처리장치는 제어장치에서 만들어진 제어신호에 근거하여 처리할 데이터와 주소 등의 오퍼랜드를 인출하고 명령어를 수행하는 장치다.

- 명령어 스트림(Instruction Stream)과 데이터 스트림(Data Stream)
  - 명령어 스트림 : 프로세서에 의해 실행되기 위하여 순서대로 나열된 명령어 코드들의 집합
  - 데이터 스트림: 명령어를 실행하는 데 필요한 순서대로 나열된 데이터 집합

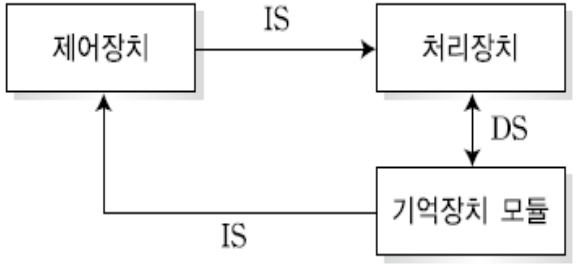
- 플린의 분류학 : 프로세서들이 처리하는 명령어 스트림과 데이터의 스트림의 수에 따라 병렬 프로세서를 분류
  - 단일 명령 스트림, 단일 데이터 스트림 : SISD
  - 단일 명령 스트림, 다중 데이터 스트림 : SIMD
  - 다중 명령 스트림, 단일 데이터 스트림 : MISD
  - 다중 명령 스트림, 다중 데이터 스트림 : MIMD

병렬 컴퓨터 프로세서 조직의 분류



SISD

- 하나의 명령어와 데이터를 순서대로 처리하는 단일 프로세서 시스템
  - 하나의 기억장치에 저장되어 있는 데이터들을 처리하기 위하여, 하나의 명령어 흐름을 순차적으로 실행한다.
  - 명령어가 실행될 때 여러 단계로 나누고 각 단계들이 중첩되면서 수행되는 명령어 파이프라이닝을 이용하여 처리 효율 향상 가능



- SISD의 동작
  - 제어장치는 기억장치에서 명령어 스트림을 인출하고, 처리장치로 보낸다.
  - 처리장치는 명령어 수행과정에서 필요한 데이터 스트림을 기억장치에서 읽고, 그 결과를 다시 기억장치에 저장한다.
  - 단일 제어장치에서 단일 명령어 스트림이 이동하고 단일 처리장치에서 단일 데이터 스트림이 이동한다.

SIMD

- 하나의 명령어 스트림(IS)이 다수의 처리장치들에서 동시 처리되는 기술
  - 각 처리장치가 각 기억장치에 저장된 독립된 데이터를 처리
  - 하나의 제어장치는 하나의 명령어를 인출하여 해독하고, 여러 개의 처리장치는 여러 데이터를 동시에 인출하여 명령어를 실행한다.
  - 벡터 프로세서와 배열 프로세서가 대표적

- 벡터 또는 배열 프로세서에서는 벡터계산을 동일 개념으로 수행

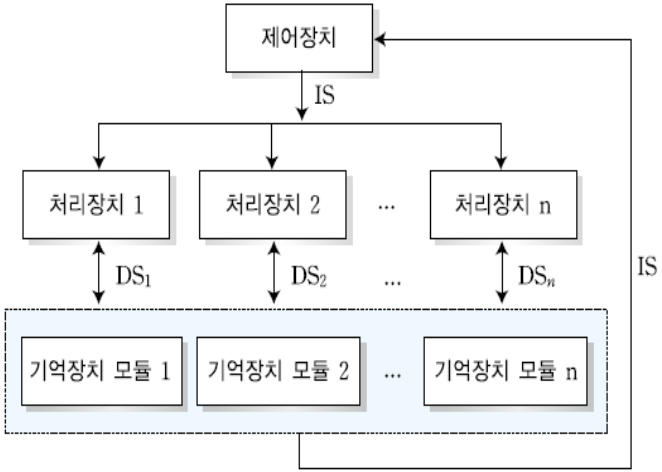
$$\begin{bmatrix} 1.5 \\ 7.1 \\ 6.9 \\ 100.5 \\ 0 \\ 59.7 \end{bmatrix} + \begin{bmatrix} 2.0 \\ 39.7 \\ 1000.003 \\ 11 \\ 21.1 \\ 19.7 \end{bmatrix} = \begin{bmatrix} 3.5 \\ 46.9 \\ 1006.903 \\ 111.5 \\ 21.1 \\ 79.4 \end{bmatrix}$$

$$A + B = C$$

  - 각각의 처리장치는 덧셈 명령에 의해서 동일한 덧셈 연산을 수행
  - 각 처리장치에서의 데이터 스트림은 서로 다른 독립된 값을 갖는다

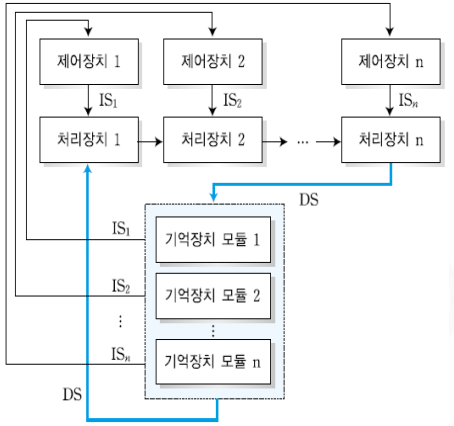
SIMD의 동작과정

- 여러 처리장치들의 동작은 모두 하나의 제어장치에 의해 통제
- 처리장치들은 독립된 기억장치 모듈을 보유한다.
- 모든 처리장치는 하나의 명령어 스트림을 실행하지만 독립된 여러 개의 데이터 스트림들을 독립적으로 동시에 처리된다.



MISD

- SIMD 구조와 반대 개념
  - 처리장치 수행 명령어는 다르지만, 전체적으로 하나의 데이터 스트림을 갖는 형태
  - 여러 제어 장치는 동시에 여러 명령어를 인출하여 각각 해독하고 하나의 처리장치는 여러 명령어를 실행하여 하나의 데이터 스트림을 갖는다.

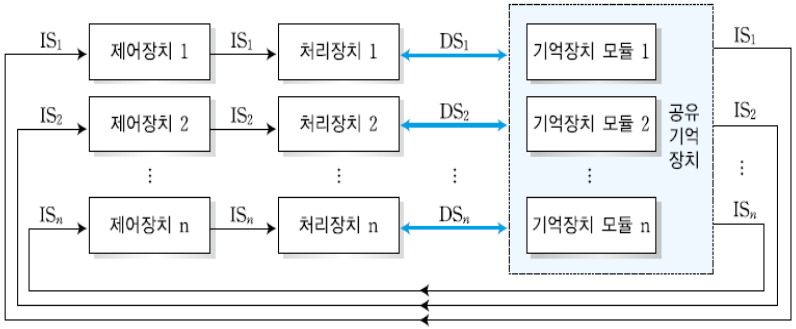


- MISD의 개념적인 동작
  - 각 처리장치에서 덧셈, 뺄셈, 곱셈, 나눗셈 명령어가 각각 실행된다면, 이 과정에서 하나의 데이터 스트림이 생성될 수는 없다.
  - MISD 구조는 비현실적으로 범용 컴퓨터 행태로 구현한 경우는 없다.

MIMD

- 일반 목적(general-purpose)의 다중 프로세서
  - 다수의 처리장치가 서로 다른 명령어들을 동시에 병렬로 실행하는 형태
  - 제어장치들은 동시에 여러 명령어를 각각 인출하고 해독하며,
    처리장치들은 여러 데이터들을 동시에 인출하여 각각 명령들을 실행한다.

- MIMD의 개념적인 동작



- 처리장치들 사이의 데이터 상호 교환 정도에 따른 분류
  - 밀접 결합(tightly coupled)형: 통신이 빈번하게 발생하는 MIMD 구조
  - 느슨 결합(loosely coupled)형: 통신의 빈도가 극히 적게 발생하는 구조

다음 시간

14주 2강. 배열 프로세서와 다중 프로세서의 개념