Java를 알고 C배우기

컴퓨터프로그래밍3

week 4-2 함수와 스택-지역변수

2022.1학기

충남대 조은선

지역 변수와 스택 프레임

```
                sum = sq_add(first, second); …
int sq_add(int first, int second)
```

sq_add( )의 스택 프레임 (활성 상태)	매개변수	first, second
	지역 변수	total
main( )의 스택 프레임 (비활성 상태)	매개변수	none
	지역 변수	first, second, sum

- 변수 명이 같아도 다른 변수
  - 서로 다른 스택 프레임에 존재
  - Sq_add의 first는 sq_add의 스택 프레임에 존재.
  - main의 first는 접근 불가
  - 매개변수는 물론 지역 변수도 마찬가지

Scope

- 변수의 속성
  - 이름, 타입, 주소, 값, 영역(scope), 존속기간(lifetime) …
- 영역(Scope)
  - 프로그램에서, 해당 변수를 사용할 수 있는 영역
  - 프로그램의 어떤 지점에서 사용할 수 있는 변수가 있으면, 그 변수는 그 지점에서 가시적(visible)이라고 한다.
  - 예) 지역변수는 선언된 내부에서는 가시적(visible)이지만, 외부에서는 가시적이지 않다

```
void func1() {
        int k = 1;
        while (1) {
                int i = 0; ... ;
                printf("%d", k);          //1을 출력
        }
        printf("%d", i); // 오류, i 가 무엇인지 모른다
}
```

Scoping 규칙

- Scoping 규칙
  - 프로그램의 어떤 지점에서 사용된 변수 이름을 어디에서 선언된 변수와 대응시켜줄 것인지를 결정하는 규칙

```
void func1() {
        int i = 0;
        while (1) {
            int k = 100;
            int i = 1; ... ;
            while (1) {
                int i = 2;
                while (1) {
                    printf("%d %d", k, i);        // 100, 2을 출력
                    // printf("%d", x);           // 없으므로 오류
                }
            }
        }
```

중첩된 구조에서 Scoping 규칙

1. 변수 이름이 사용된 block이나 함수에서 선언되어 있으면 그대로 사용
2. 선언이 없으면, 둘러싸고 있는 block이나 함수로 이동해서 선언을 찾는다. 있으면 그대로 사용
3. 선언이 없으면, 또 다시 둘러싸고 있는 block이나 함수로 이동해서 선언을 찾아보고, 있으면 그대로 사용.. 반복
4. 전역변수 선언까지 도착했는데 해당 변수가 없으면?
   ➔ 오류!

존속기간 (Lifetime)

```
auto int sum;
```

- 변수 영역(Scope, Visibility)은 공간 개념
- 변수 수명(Lifetime, Extent)은 시간 개념
- 지역 변수(매개변수)는 함수 호출 시에 생성, 함수 종료 시에 소멸
- 지역 변수는 자동 변수
  - 스택 프레임의 생성과 소멸에 따라 자동으로 수명이 결정됨
  - 대부분 지역 변수가 자동 변수이기 때문에 auto를 생략

- 전역 변수의 수명
  - 프로그램이 시작할 때부터 끝날 때까지
  - 지역 변수는 스택 프레임에 저장. 스택 프레임이 팝 되면 소멸
  - 전역 변수는 데이터 세그먼트라는 고정 메모리에 생성
  - 지역 변수는 스택 프레임에 남아 있던 쓰레기 값으로 초기화
  - 전역 변수는 프로그램 시작과 함께 자동으로 0으로 초기화

여러가지 변수의 Scope과 Lifetime

- 전역변수
  - Scope: 전체 프로그램
  - Lifetime: 프로그램 시작부터 끝까지
- 지역변수
  - Scope: 해당 함수
  - Lifetime: 해당함수가 호출되어 실행될 때 부터, 리턴될 때 까지
- 정적(static) 변수
  - Scope: 해당함수 ..... 지역변수와 동일
  - Lifetime: 해당함수가 처음 호출되어 실행될 때 부터 프로그램 끝날 때 까지 ... 전역변수와 유사

레지스터 변수

```
void add( ){
    int i, j;
    register int count = 0;

    for (i = 0; i < MAX; i++)
        for (j = 0; j < MAX; j++)
            count++;
}
```

- 변수를 CPU 레지스터에 저장해 달라는 요구
- WHY? 메모리에 비해 빠른 접근 속도
- 컴파일러 스스로 최적화에 의해 레지스터를 할당하기도 함
- 모든 레지스터가 사용 중이면 할당 받지 못할 수도 있음

volatile

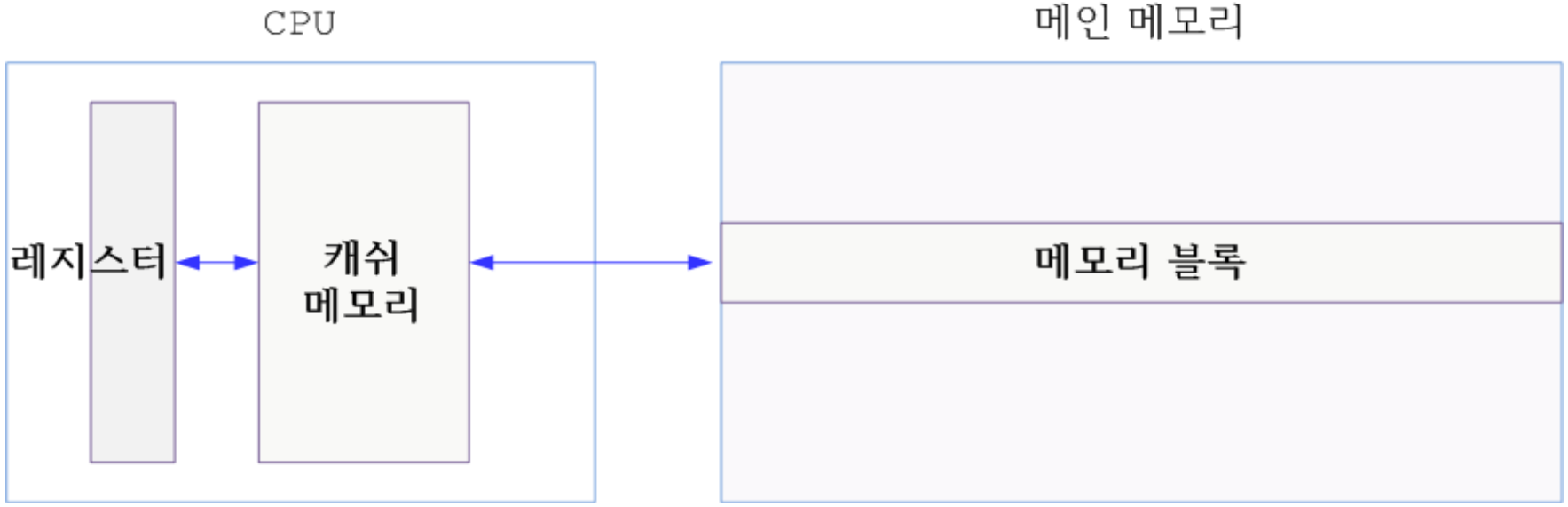
```
int ready = 0;
while (!ready){
    printf("Wating for the ready signal.\n");
}
```

- 프로그래머
  - 반복문 실행 도중에 외부 코드가 ready를 1로 바꾸어 줄 것을 기대하고 위 코드를 작성 (누가?)
- 컴파일러
  - while (1)로 최적화(무한 루프)

```
volatile int ready = 0;
```

- volatile은 컴파일러 최적화 대상에서 제외

volatile



- Cache Memory
  - 고속 메모리.
  - 캐시 크기(32KB ~ 8MB), 블록 단위(32 - 128 Bytes)로 저장
  - 캐쉬 메모리로 옮긴 이후 메인 메모리 변수 값이 바뀔 경우가 문제. 그 경우 CPU는 바뀌기 이전 상태의 변수 값을 읽은 것이 됨

- Volatile은 접근이 필요한 바로 그 시점에 메인 메모리에 무조건 직접 접근하라는 뜻

```
volatile int ready = 0;
```

Quiz

- 다음 중 가장 거리가 먼 것을 모두 고르시오.

  (1) 지역변수와 전역변수가 동일한 이름인 경우 지역변수를 따른다.

  (2) volatile은 캐시가 되어 있어도 무조건 메모리에서 읽으라는 뜻이다.

  (3) register는 레지스터에 변수를 할당하라는 뜻이다.

  (4) 지역변수의 lifetime은, 자신을 정의한 함수가 리턴되면 종료된다.

  (5) static 변수는 scope은 전역변수와 유사하고 lifetime은 지역변수와 유사하다.