소프트웨어 공학개론

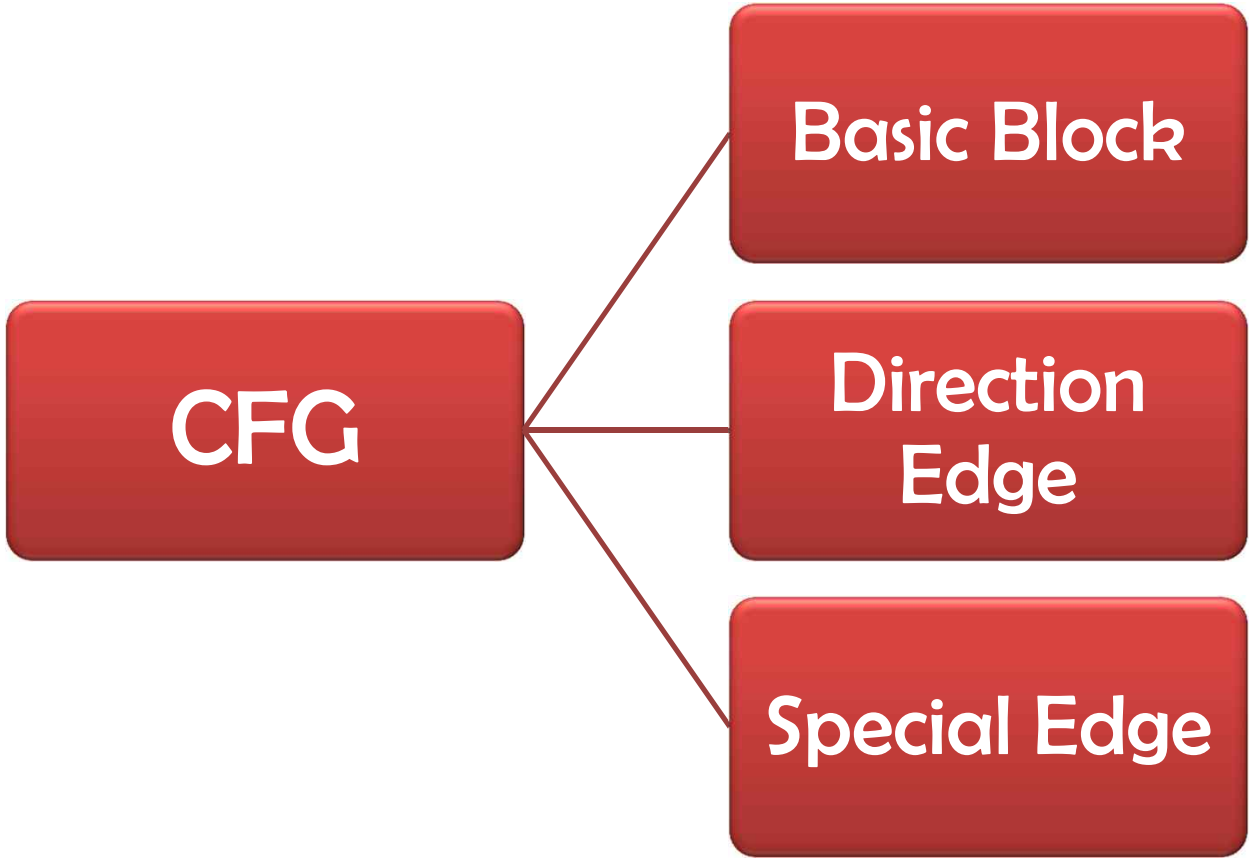
CFG(Control Flow Graph)

201011318 김슬기

201011334 박진성

What Is CFG?

- CFG – Control Flow Graph



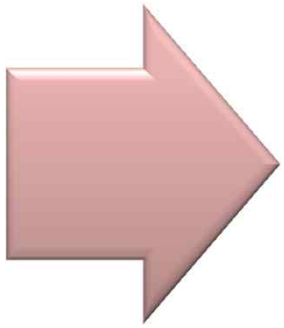
What Is CFG?

- Basic Block

- **Basic Block**은 하나의 일을 하는 코드 조각을 **Basic Block**으로 묶어서, **Edge**들로 연결 시키게 한다.

What Is CFG?

- Direction Edge



- **Directed Edge**는 하나의 **jump target**을 나타낸다.
  어떤 **BB**에서 또 다른 **BB**로 점프 할 때 쓰인다.

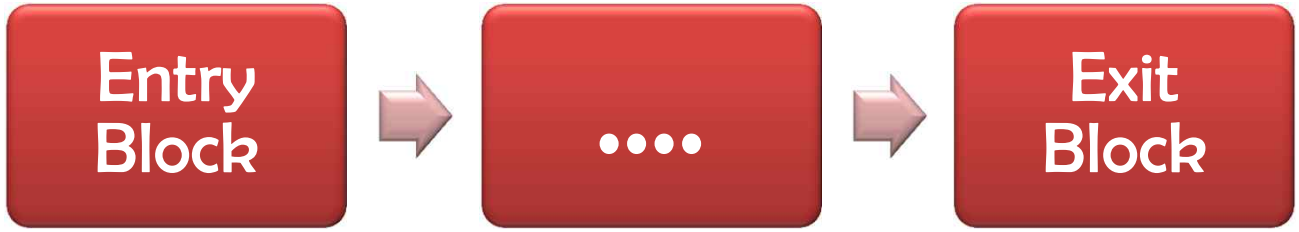
What Is CFG?

• Special Edge

- 인공적인 종류의 **Edge**를 소개할 때 사용하거나, 일반적인 **Edge**와는 다르게 진행되는 종류의 **Edge**를 소개하는데 필요하다.

- **Special Edge**의 종류로는 **Back Edge, Critical Edge, Abnormal Edge, Impossible Edge**가 있다.

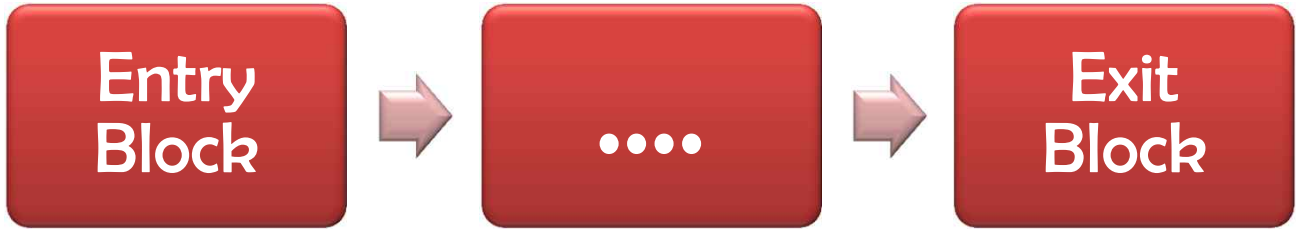
What Is CFG?



- CFG는 두 개의 특별한 Basic Block들이 있다.

- Entry Block은 Code의 시작을 의미
- Exit Block은 Code의 끝을 의미

What Is CFG?



- **Compiler**에서 많이 쓰이는 자료구조

- **Code** 조각들을 **BB(Basic Block) Node**에 담아 **Edge**로 연결시킨 그래프
- **CFG**의 그래프를 **DFS**로 탐색하여 그대로 컴파일을 할 수도 있다.

What Is CFG?

- C언어 같은 경우, 파싱하여 그대로 어셈블러의 BBNode들을 결합하여 CFG 자료구조를 만든 후,

- 그들을 DFS로 탐색하며, 다시 어셈블리 코드로 만들어 내면 컴파일이 완료되게 된다.

CFG Algorithm

- 먼저 **BB Node**들을 뽑아내야 한다.

- **Assembly**에서 자주 사용되는 아주 기본적인 **BB Node**는 **jmp(점프), cmp(비교), mov(복사)** 등이 있다.

- 이들을 조합하여 **C**언어를 분석하여 **CFG**로 만들면 된다.

CFG Algorithm

-example1.c-

```
int d;                          /*step 1*/
printf("number : ");            /*step 2*/
scanf("%d", &d);    /*step 3*/
printf("%d\n", d);; /*step 4*/
```

- 각각의 step을 하나의 BB로 본다.
- for문과 같은 loop가 없으므로, 아래로 천천히 내려가면서 수행하게 된다.
- 이때, 각 함수들을 다시 하나의 BB로 보게 된다.

CFG Algorithm

Step 1

4 byte만큼 load한다.

Step 2

printf를 메모리에 올리고, 그 주소로 점프하여 실행한다.

Step 3

scanf를 메모리에 올리고, 그 주소로 점프하여 실행한다.

Step 4

printf를 메모리에 올리고, 그 주소로 점프하여 실행한다.

CFG Algorithm

- **printf, scanf** – 이미 **Compile**된 **Binary**
  - **Memory**에 올려 실행하기만 하면 된다.

- 사용자가 만든 임의의 함수
  - 함수 자체가 **CFG**를 가지고, 그 **CFG**를 탐색하여 **Assembly**로 만든다.
  - 이것도 하나의 **BB Node**가 될 수 있다.

CFG Algorithm

- Loop를 갖는 경우

-example2.c-

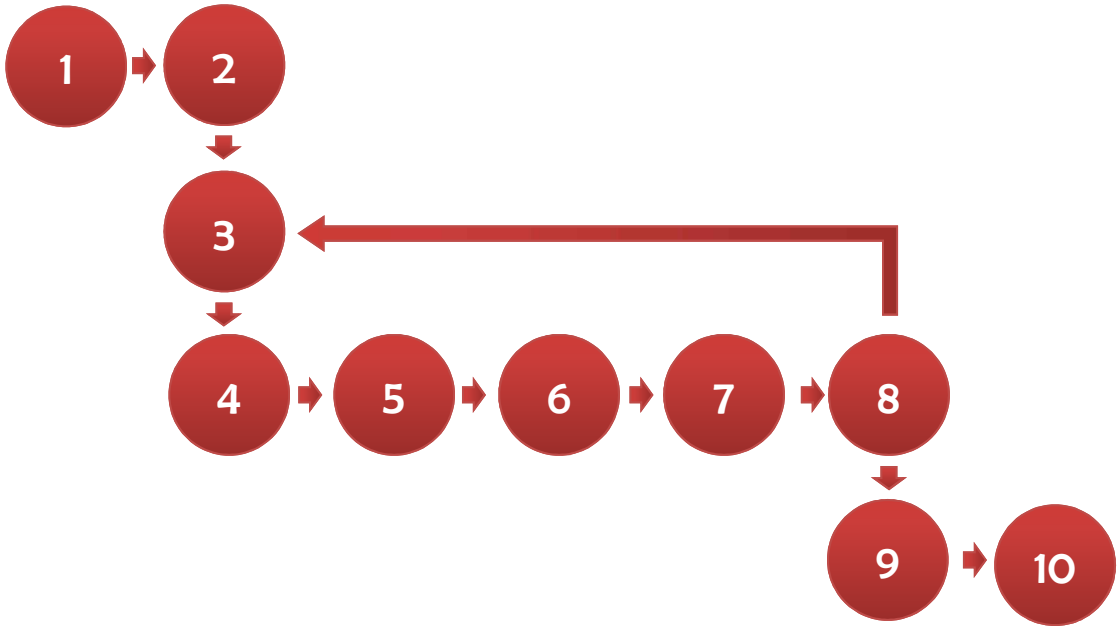
```
1. int a;
2. for(int i = 0; i<10; ++i) {
3.       a = i;
4. }
```

- 이런 코드를 그래프로 그리기 위해서는 약간의 수정이 필요하다.

CFG Algorithm

```
1.  int a;
2.  int i = 0;
3.  loop1:
4.  if(i<10) {
5.      a = i;
6.      ++i;
7.      goto loop1;
8.  } else {
9.      ;
10. }
```

- 옆과 같이 수정하면,
  그래프가 분기(if, else)를 갖게 된다.



- 위와 같은 원활한 BB를 만들 수 있게 된다.

CFG Algorithm을 만드는 Algorithm에 대한 Statement of Purpose

- First
  - Code를 집어 넣으면, 자동으로 CFG를 만들어 출력해야 한다.
- Second
  - Code에서 Loop가 감지되면, CFG는 Code 또는 Assembly를 수정하여, CFG에 약간의 수정을 가해줘야 한다.

CFG Algorithm을 만드는 Algorithm에 대한 Statement of Purpose

- Third
  - 분기(If, else)가 감지 되면, DFS 탐색할 때, if가 먼저 탐색 되도록 왼쪽에 두면서 분기시켜줘야 한다.
- Forth
  - BB내부에 Data는 Assembly Code 조각뿐만 아니라 BB 자신도 될 수 있어야 한다.
- Fifth
  - 문자열 Reader는 전적으로 외부 Interface에 의존한다.

Q & A