60200307 우연정

목차

1. Process Memory Layout

코드가 실행되었을 때 메모리 레이아웃이 어떻게 변하는rk? 컴파일러가 두 개로 나눠서 코드를 해석한다. 하나는 머신이 실행할 수 있는 executable한 코드와 cpu가 실행하기 위해서 컴파일러가 준비하는 코드로 나뉜다. 이 두 개는 다르다. 즉 실행코드를 만들기 위한 정보와 실행코드 그 자체는 다른 것이다.

cpu 명령어는 크게 세 가지로 나누어지는데, 메모리에서 데이터를 읽어오거나 쓰는 것, 계산하는 것, 코드의 흐름을 바꾸는 것(flow control, 흐름 제어)이 있다. flow control은 코드가 실행되는 순서를 바꾸는 것이다. (예) if, while, 함수 호출)

<table>
<tr>
<td>

```
public class CMain {
        static private int s;
        private a, b;
        private void run() {
                View view;
                int x;
                x = a+ b;
                view = new View();
                view.showUserInfo();
        }

public static void main(String[] args) {
                int x;
                CMain main;
                main = new CMain();
                main.run();
        }
}
```

</td>
<td>-> 메모리를 확보하라는 명령어

-> 실행코드. cpu가 실행함. cpu가 계산을 하거나 점프를 하거나 메모리를 읽는다.
new는 메모리를 만드는 명령어임에도 불구하고 실행문이다. 일반적으로 메모리를 만드는 것은 컴파일러가 저장공간을 확보하고 위치만 알고 있을 뿐 실제로 만들지는 않는다. 그런데 이 new 명령어는 독특하게 힙에 실시간으로 메모리를 만든다. (실행 시 만든다.)</td>
</tr>
<tr>
<td>실행할 수 있는 명령문</td>
<td rowspan="2">실행문들은 함수 안에 들어가 있는 명령어들을 말한다.
main()과 run()이 머신 코드로 바뀔 수 있는 명령어의 집합인 것이다.
이 부분이 코드 세그먼트로 들어오고, 기계어로 바뀐다.
클래스 안에 있어도 executable한 명령어가 있고, 정의(declaration)이 있다. 즉 문장과 그 문장을 실행하기 위한 부가 정보로 이루어졌다.</td>
</tr>
<tr>
<td>

```
private void run() {
        x = a+ b; //계산
        view = new View(); //힙메모리 할당
        view.showUserInfo(); //함수 호출(점프)
        }

public static void main(String[] args) {
        main = new CMain();
        main.run();
        }
}
```

</td>
</tr>
</table>

데이터 세그먼트, 힙 세그먼트, 스택 세그먼트는 코드 세그먼트가 쓰는 데이터 영역이다. 규칙은 코드 세그먼트에 쓰여있는 것이다. 이는 그 데이터를 쓸 때 어떤 값을 읽어서 어떤 값을 쓸지(계산)를 결정하는 규칙이다. 코드 세그먼트에 쓰여 있는 이름들은 각 세그먼트에서의 데이터 주소를 의미한다.

힙 세그먼트만 런타임에 시피유 명령어로 실행이 된다. new라는 명령어는 아예 동적으로 실행되게 되어있다.

흐름을 바꾸는 명령어는 코드 세그먼트에 있는 주소로 점프하는 것이고, 계산하는 명령어는 데이터가 있는 주소로 가서 데이터를 읽어서 규칙에 의해서 계산을 해서 다시 쓰는 것이다.

public class CMain {
static private int s;
private a, b;
private void run() {
View view;
int x;
x = a+ b;
view = new View();
view.showUserInfo();
}
 	
public static void main(String[] args) {
int x;
CMain main;
main = new CMain();
main.run();
}
}
	-> 데이터, Declaration, non executable, 실행 가능한 명령어가 아니다. 컴파일러가 이름을 만들어서 메모리를 할당하기 위한 명령어들. 컴파일러가 미리 만들어내는 것이다.

2. Memory Allocation

자바에서 메모리를 할당할 때 포인터와 객체를 만들고, 포인터에 객체의 주소를 할당하는 방식으로 한다(binding). 예를 들면, new Student() 라는 코드는 힙에 메모리를 할당하라는 뜻이다. 기계어로 바꾸면 그 뒤에는 파라미터로 사이즈가 들어간다. (기계어로 object = malloc(16)) 그리고 Student클래스 안에 있는 구조(construct)를 호출한다. (object.student()) 따라서 new Student()는 실제 명령어로는 두 줄인 것이다. 즉 new라는 명령어는 어떠한 사이즈만큼 메모리를 allocation 하고 클래스 안에 있는 어떠한 함수를 호출하라는 뜻이다. 이는 동적 allocation에 관한 것이며 일반적인 메모리 allocation이다. 오브젝트 타입인 경우이다.

Primitive 데이터 타입(int, float, boolean)의 경우 객체가 아니라 자바가 원래부터 가지고 있는 데이터 타입들이다. 예를 들어 'int x'를 하면 데이터가 바로 만들어진다. 객체와 포인터를 분리하지 않는다. 객체의 구조를 갖고 있지 않으며, 그것 스스로가 데이터인 것이다.
객체는 클래스가 만들어져야만 만들 수 있다. 오브젝트는 클래스의 인스턴스(실체)이다.

cpu의 첫 번째 명령어는 계산이다. 두 번째 명령어는 컨트롤 유닛이 하는 흐름 제어,
세 번째는 cpu가 메모리에서 특정 주소에 있는 데이터를 읽어오고 쓰는 명령어(load, store)이다. 네 번째는 IO interrupt이다. IO 디바이스는 메모리의 일부이다. 모니터와 키보드, USB를 말하며, 명령어는 쓰는 것과 읽는 것 두 가지가 있다. 계산하는 과정에 읽고 쓰는 것이 포함되어있다.

```
public class CMain {
	static private int s;
	private a, b;
	private void run() {
		View view;
		int x;
		x = a+ b;
		view = new View();
		view.showUserInfo();
	}

public static void main(String[] args) {
		int x;
		CMain main;
		main = new CMain();
		main.run();
	}
}
```

위 코드를 실행하면 x는 데이터 세그먼트에 만들어진다. main도 static에 만들어진다. new Cmain()을 하면 힙 세그먼트에 만들어진다.

프로그램이 시작되는 순간 x와 main은 이미 만들어져서 데이터 세그먼트에 들어와있고, 첫 번째 줄을 실행하는 순간 main에 힙의 주소(new CMain()의 주소)를 집어넣는다. main이 힙을 가리키는 포인터가 된다. new를 하면서 만들어진 주소가 500번이면, main에 500이 들어간다.

두 번째 줄 main.run()을 하면 데이터 세그먼트에는 x, main이 그대로 있고 s가 들어간다. 그리고 힙 세그먼트에 a와 b가 들어간다.(new CMain()과 같은 뜻) 스택 세그먼트에 view와 x가 만들어진다. 스택은 함수가 끝나면 없어진다. 스택에 생기는 것은 local variable만 생기는 것이 아니라 return address, return parameter, return type 등도 들어간다.