컴파일러 11주차 실습 보고서

201702086 한정경

```
.class public Main2
.super java/lang/Object
; strandard initializer
.method public <init>()V
aload_0
invokenonvirtual java/lang/Object/<init>()V
return
.end method
```

그림 1 Main2 초기화

그림 1은 Main.j 파일의 시작 부분을 나타낸다. 먼저 파일명이 Main2인 클래스를 초기화한다. `.class public Main2`에서 public으로 선언된 Main2의 이름을 가진 클래스를 정의한다. `.super java/lang/Object`에서 Obejct클래스를 상속받음을 선언한다. `.method public <init>()V`에서 return이 void임을 정의한다.

```
; Sum 함수 시작
.method public static Sum(I)I   ; I 받아 I 리턴
.limit stack 32
.limit locals 32

iload_0                         ; n값(100)

iconst_1                        ; result 초기화
ldc 0                           ; result 값 0 스택에 저장
istore_1
```

그림 2 Sum 함수 1

그림 2는 1부터 100까지의 함을 구하는 함수 Sum의 시작 부분을 나타낸다. 인자와 반환 타입, 스택을 설정한다. `iconst_1, istore_1`을 통해 합을 저장할 공간을 0으로 초기화한다.

```
iconst_2                    ;i 초기화
ldc 1
istore_2                    ; i에 1 저장
1: iload_2                  ; i 불러옴
bipush 100                  ; 100과 i 비교
if_icmpgt 2                 ; 조건에 맞지 않으면 출력으로
iload_1                     ; load result
iload_2                     ; load i
iadd                        ; result + i
istore_1                    ; result = result + i
iinc 2 1                    ; i ++
goto 1                      ; 반복
```

그림 3 Sum 함수 2

그림 3은 Sum 함수에서 반복을 통해 1부터 100까지의 합을 구하는 반복문의 코드를 나타낸다. 반복문을 시작하기 위해 `iconst_2`와 `istore_2`로 i를 1로 초기화한다. `bipush 100`을 통해 i와 정수값 100을 비교한다. 비교 결과를 바탕으로 `if_icmpgt` 에서 goto 이동 여부를 결정한다. 즉, i > 100 이라면 설정해둔 2: 로 넘어가고 그렇지 않다면 아래의 작업을 수행한다. i <= 100 이라면 `iadd` 로 `iconst_1`에 i의 값을 더한다. `iconst_1` 변수의 이름을 result라고 가정했을 때, result += i 의 작업이 수행된다. `iinc 2 1` 로 i가 저장된 2에 1의 값을 추가한다. 마지막으로 `goto 1` 을 통해 i를 불러오는 `iload_2`를 수행하여 반복문을 계속할 것인지를 판별한다.

```
2: iload_1          ; result 불러옴
ireturn           ; result 반환
.end method       ; Sum 함수 종료
```

그림 4 Sum 함수 3

그림 4는 Sum 함수의 반환 부분을 나타낸다. 결과값이 저장된 1을 불러와 반환하고 함수를 종료한다.

```
; Main2 함수 시작
.method public static main([Ljava/lang/String;)V
.limit stack 32
.limit locals 32


getstatic java/lang/System/out Ljava/io/PrintStream;

ldc 100             ; num의 초기값 0
istore_2            ; 0 저장
iload_2             ; 100 불러와 인자로 넘김

invokestatic Main2/Sum(I)I      ; 모두 더한 값 가져옴
invokevirtual java/io/PrintStream/println(I)V   ; 값 출력
return
.end method
```

그림 5 Main2 함수

그림 5는 Main2 함수를 나타낸다. `idc 100`에서 Sum의 인자로 100을 전달한다. Sum함수의 결과값을 `println(I)V`로 출력한다.

```
han@hanjeong-gyeong-ui-MacBookAir jasmin-2.4 % java -jar jasmin.jar Main2.j
Generated: Main2.class
```

Main2.j 파일을 jasmin-2.4의 경로에 추가한 뒤 명령어를 사용하여 Main2.class 파일을 생성한다.

```
han@hanjeong-gyeong-ui-MacBookAir jasmin-2.4 % java Main2
5050
```

Main2를 실행하여 1부터 100까지의 결과인 5050이 나오는 것을 확인할 수 있다.

후기

이론 시간에 배운 것을 활용할 수 있었던, 좋은 실습이었습니다. 코드가 어떻게 어셈블리로 변하는지 보는 것이 재미있었습니다. 강의와 크게 연결되는지는 모르겠지만 역어셈블에 대해서도 가볍게 이야기 하면 재밌을 것 같습니다. 감사합니다

Main2.j 의 전체 코드는 다음과 같다

```
.class public Main2
.super java/lang/Object
; strandard initializer
.method public <init>()V
aload_0
invokenonvirtual java/lang/Object/<init>()V
return
.end method

; Sum 함수 시작
.method public static Sum(I)I   ; I 받아 I 리턴
.limit stack 32
.limit locals 32

iload_0                         ; n 값(100)

iconst_1                        ; result 초기화
ldc 0                           ; result 값 0 스택에 저장
istore_1

;################################
;########   반복분 코드 ########

iconst_2                    ;i 초기화
ldc 1
istore_2                    ; i 에 1 저장
1: iload_2                  ; i 불러옴
bipush 100                  ; 100 과 i 비교
if_icmpgt 2                 ; 조건에 맞지 않으면 출력으로 넘어감   i > 100 일 경우
iload_1                     ; load result
iload_2                     ; load i
iadd                        ; result + i
istore_1                    ; result = result + i
iinc 2 1                    ; i ++
goto 1                      ; 반복

;################################
;################################

2: iload_1          ; result 불러옴
ireturn         ; result 반환
.end method     ; Sum 함수 종료

; Main2 함수 시작
.method public static main([Ljava/lang/String;)V
.limit stack 32
.limit locals 32


getstatic java/lang/System/out Ljava/io/PrintStream;

ldc 100             ; num 의 초기값 0
istore_2            ; 0 저장
iload_2             ; 100 불러와 인자로 넘김

invokestatic Main2/Sum(I)I      ; 모두 더한 값 가져옴
invokevirtual java/io/PrintStream/println(I)V   ; 값 출력
return
.end method
```