프로젝트 #1-c

20181259

조수민

130

프로젝트 #1-c

<가>

1) [BUG FIX] immediate, indirect 연산이 수행되지 않던 버그

버그 상황

```
T00001D0100000F200A4B1000003E2000454F4610
```

```
LDA #3  SET LENGTH = 3
```

버그 발생 조건

operand 앞에 문자 # 혹은 @가 존재할 때 해당 문자를 자르고 Token 에 저장했기 때문에 문제가 발생했음

버그 수정 방식

앞의 문자를 포함한 모든 문자를 Token 으로 저장한 후 Pass 1 에서 판별

버그 수정 이후

```
T0000001D1720274B10000003202329000033200 74B1000003F2FEC0320160F2016
```

2) [BUG FIX] 오브젝트 코드에서 텍스트의 메타데이터가 출력되지 않던 버그

버그 상황

```
T0000001720274B10000003202329000033200 74B1000003F2FEC032
T00001D0100000F200A4B1000003E2000454F4610
```

버그 발생 조건

text 레코드의 메타 정보 ex) 코드 시작 주소, 코드 길이 등을 따로 저장하지 않고 ToString을 통해 즉석에서 출력하기 때문

버그 수정 방식

여전히 text 레코드의 메타 데이터를 따로 저장하지 않음. 하지만 먼저 길이를 체크한 후 출력하도록 ToString을 수정하고, Token 에 해당하는 locctr의 값을 저장하는 address 멤버 변수를 추가하여 해당 데이터를 처음에 출력하는 방식으로 변경

버그 수정 이후

```
T0000001D1720274B1000000320232900003320074B1000003F2FEC0320160F2016
T00001D100100030F200A4B1000003E2000454F46
```

3) [미구현 기능 지원] Word의 연산자 지원(+,-) 및 Modification Record 반영

기능 설명

```
HRDREC 00000000002B
RBUFFERLENGTHBUFEND
T000000B410B400B44077201FE3201B332FFADB2015A004332009571
T00001D3B2FE9131000004F0000F10B
M00001805+BUFFER
M00002105+LENGTH
E
```

```
MAXLEN   WORD     BUFEND-BUFFER
```

심볼에 연산자가 존재하면 인식하지 못하고, Modification Record에도 반영되지 않음

기능 구현 방식

Directive 명령어일 때 심볼에 연산자가 포함되어 있는 경우, 연산자를 기준으로 operand를 split 한 후 ref symbol에 있는 operand일 시 Modification에 반영 +,- 두개를 연산자로 정하여 두가지를 구현

기능 동작 상황

```
M00001805+BUFFER
M00002105+LENGTH
M00002806+BUFEND
M00002806-BUFFER
E
```

4) [미구현 기능 지원] 잘못된 심볼을 발견하면 에러

기능 설명

```java
public static void main(String[] args) {
    try {
        Assembler assembler = new Assembler(instFile:"프로젝트폴더/inst_table.txt");
        ArrayList<String> input = assembler.readInputFromFile(inputFileName:"프로젝트폴더/input2_error.txt");
        ArrayList<ArrayList<String>> dividedInput = assembler.divideInput(input);
```

에러인 input을 넣고 돌려도

```
cd /Users/josumin/Downloads/\[130\]_프로젝트1b_조수민_20181259/source ; /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-20.jdk/Contents/Home/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/josumin/Library/Application\ Support/Code/User/workspaceStorage/eca8886ada1e5ead529fa4c159c58861/redhat.java/jdt_ws/source_6fbb4f0d/bin Assembler
```

정상적으로 실행되고

```
HCOPY  000000001033
DBUFFER000033BUFEND001033LENGTH00002D
RRDREC WRREC
T0000001720274B10000003202329000033200074B1000003F2FEC0320160F2016
T00001D0100000F200A4B1000003E2000454F4610
M00000405+RDREC
M00001105+WAAAA
M00002405+WRREC
E
```

터무니없는 오브젝트 코드를 생성한다

기능 구현 방식

기존에는

1. 심볼이 심볼테이블에 존재하는지 않아도 에러를 throw 하지 않았다.
2. 명령어 형식이 4형식이면 REF로 판단하고 00000의 주소값을 반환했다.

기능 구현 이후

Pass 2 에서 심볼이 나오면 심볼테이블에 있는 지 검사한 후 없다면 REF 테이블에 있는 지 검사.

둘 다 존재하지 않을 시 ERROR를 Throw 했다.

이는 Extended된 명령어에도 동일하게 적용된다.

기능 동작 상황

```
cd /Users/josumin/Desktop/sp/SP24_PRO1B ; /usr/bin/env /Library/
677774f08532f85c4bdb6b9/redhat.java/jdt_ws/SP24_PRO1B_9555fbc3/bin
java.lang.RuntimeException: undefined symbol in expression: WAAAA
```

undefined symbol : WAAAA 라는 에러를 띄우게 된다.

총평

사실 항목으로 나누긴 했지만,

사실 아예 잘못된 오브젝트 코드를 정답(혹은 정답에 가까운) 오브젝트 코드로 바꾼 작업이라고 생각합니다.

제가 예시와 다르게 구현한 부분은

1. LTORG가 존재해도 줄바꿈 하지 않고 이어서 출력
2. LTORG가 없이 리터럴이 존재할 시 오브젝트 코드에 반영하지 않음

사실 2번같은 경우에는 제가 구현에 실패한 것에 가깝긴 합니다. 하지만 나름의 이유를 쓰자면 LTORG가 없이 리터럴을 사용할 이유가 없다고 판단하여 구현하지 않았습니다. RSUB 이후에 LTORG를 써준다면, 예시로 주어진 오브젝트 코드와 동일한 형태로 출력이

됩니다.