시스템 프로그래밍 개별 프로젝트 #2

1. 프로젝트 문제 및 목표

프로젝트 #1에서 구현한 셸(shell)에 assemble 기능을 추가하는 프로그램. SIC/XE의 assembly program source 파일을 입력 받아서 object파일을 생성하고, 어셈블리 과정 중 생성된 symbol table과 결과물인 object 파일을 볼 수 있는 기능을 제공해야 함. 교재의 2.2까지 설명된 SIC/XE 어셈블러의 기능을 구현함을 원칙으로 한다.

2. 요구사항

2.1 프로젝트 목표 설정

- 이미 제출한 프로젝트#1에 아래의 기능들을 추가한다.
- 구현해야 할 사항들 (다음 페이지에 추가 설명 제공.)
  - ① Shell 관련 명령어들 (help, type)
  - ② assembler (assemble)
  - ③ assemble 관련 명령어 (symbol)

2.2 합성

프로젝트 #1에서 구현한 셸(shell)에 assemble 기능을 추가하는 프로그램을 작성하는 프로젝트로, SIC/XE machine의 assembly program source 파일을 입력 받아서 object파일을 생성하고, 어셈블 과정 중 생성된 symbol table과 결과물인 object 파일을 볼 수 있는 기능을 제공해야 한다. 이와 같은 기능을 제공하는 프로그램을 작성하기 위해 필요한 자료구조와 알고리즘을 구상하여 전체적인 프로그램을 설계한다.

2.3 제작 / 2.4 시험 / 2.5 평가

1) Shell 관련 명령어

① **sicsim> help**

- 아래와 같이 Shell에서 실행 가능한 모든 명령어들의 리스트를 화면에 출력해준다.

```
sicsim> help
h[elp]
d[ir]
q[uit]
hi[story]
du[mp] [start, end]
e[dit] address, value
f[ill] start, end, value
reset
opcode mnemonic
opcodelist
assemble filename
type filename
symbol
```

② sicsim> **type filename**

- filename에 해당하는 파일을 현재 디렉터리에서 읽어서 화면에 출력한다.
- 현재 디렉터리에 해당 파일이 존재하지 않으면 에러 메시지를 출력한다.
- filename이 디렉토리인 경우는 고려하지 않는다.
- 시스템 콜을 사용하지 않는다.

ex) sicsim> type copy.obj

```
sicsim> type copy.obj
HCOPY  000000001077
T0000001D17202D69202D4B10103603202629000033200 74B10105D3F2FEC032010
T00001D130F20160100030F200D4B10105D3E2003454F46
T0010361DB410B400B44075101000E32019332FFADB2013A00433200857C003B850
T0010531D3B2FEA1340004F0000F1B410774000E32011332FFA53C003DF2008B850
T001070073B2FEF4F000005
```

2) SIC/XE 어셈블러 명령

시스템 프로그래밍 프로젝트 #2

① **sicsim> assemble filename**

- filename에 해당하는 소스 파일을 읽어서 object파일과 리스팅 파일을 만든다.
- 소스 파일의 확장자는 .asm 이다.
- 리스팅 파일의 파일명은 소스 파일과 동일하고 확장자는 .lst 이다.
- object 파일의 파일명은 소스 파일과 동일하고 확장자는 .obj 이다.
- 소스파일에 에러가 존재할 경우, 리스팅 파일과 object파일을 생성하지 않고 에러 내용을 화면에 출력한다. 에러 발생시 바로 명령이 종료된다.
- 에러의 내용은 디버깅을 위해 어떤 라인에서 에러가 발생했는지 출력한다. (에러문구 없이 .lst,.obj파일을 생성하지 않으면 0점)
- 라인번호는 5의 배수 단위로 지정하여 출력한다.
- 각 문자열의 길이는 30자 이내로 가정한다.
- label은 영문, 숫자를 input으로 가정하며, 그 외 특수문자는 고려하지 않는다. (맨 앞에는 숫자 불가)

<명령어 예시>

```
sicsim> assemble 2_5.asm
[2_5.lst], [2_5.obj]
```

<u>시스템 프로그래밍 프로젝트 #2</u>

```
sicsim> type 2_5.obj
HCOPY  000000001077
T0000001D17202D69202D4B10103603202629000033200074B10105D3F2FEC032010
T00001D130F20160100030F200D4B10105D3E2003454F46
T0010361DB410B400B44075101000E32019332FFADB2013A00433200857C003B850
T0010531D3B2FEA1340004F0000F1B410774000E32011332FFA53C003DF2008B850
T001070073B2FEF4F000005
M00000705
M00001405
M00002705
E000000
```

```
sicsim> type 2_5.lst
5       0000    COPY    START   0
10      0000    FIRST   STL     RETADR          17202D
15      0003            LDB     #LENGTH         69202D
20                      BASE    LENGTH
25      0006    CLOOP   +JSUB   RDREC           4B101036
30      000A            LDA     LENGTH          032026
35      000D            COMP    #0              290000
40      0010            JEQ     ENDFIL          332007
45      0013            +JSUB   WRREC           4B10105D
50      0017            J       CLOOP           3F2FEC
55      001A    ENDFIL  LDA     EOF             032010
60      001D            STA     BUFFER          0F2016
65      0020            LDA     #3              010003
70      0023            STA     LENGTH          0F200D
75      0026            +JSUB   WRREC           4B10105D
80      002A            J       @RETADR         3E2003
85      002D    EOF     BYTE    C'EOF'          454F46
```

(생략)

```
210     105D    WRREC   CLEAR   X               B410
215     105F            LDT     LENGTH          774000
220     1062    WLOOP   TD      OUTPUT          E32011
225     1065            JEQ     WLOOP           332FFA
230     1068            LDCH    BUFFER, X       53C003
235     106B            WD      OUTPUT          DF2008
240     106E            TIXR    T               B850
245     1070            JLT     WLOOP           3B2FEF
250     1073            RSUB                    4F0000
255     1076    OUTPUT  BYTE    X'05'           05
260                     END     FIRST
```

자세한 내용은 책 2.2의 figure 2.5참고(3rd edition 기준)

② **sicsim> symbol**

- assemble 과정 중에 생성된 symbol table을 화면에 출력합니다. Symbol table 은 각자 설계를 하고, 출력은 아래와 같이 한다

(출력형식을 꼭 지킬 것)

- 가장 최근에 assemble 한 파일의 symbol table을 출력한다.

- symbol의 출력은 symbol을 기준으로 알파벳 내림차순으로 정렬이 되어야 한다. (내림차순 정렬 순서 지킬것)

sicsim> assemble 2_5.asm

[2_5.lst], [2_5.obj]

sicsim> symbol

(₩t)RETADR(₩t)0030

=>실제 출력 될 시에는

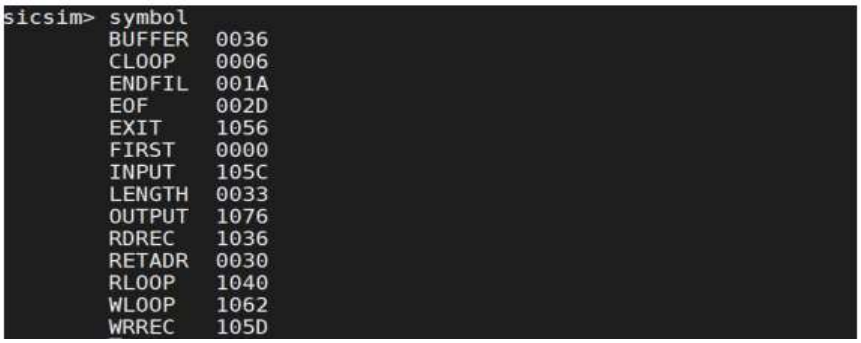
RETADR 0030

-즉 하나의 symbol당 한 line을 차지하고, 탭+Symbol+탭+주소값+₩n

을 의미한다. 맨 마지막 줄에는 ₩n(엔터)를 빼준다.

(명령어 실행 뒤 sicsim>이 다시 나올 때는 정상적으로 개행되어 나와야 한다)

<출력 예시>



교재의 2.2까지 설명된 SIC/XE 어셈블러의 기능을 구현함을 원칙으로 한다.

원래의 SIC/XE machine은 standard machine에 하위 호환 되어야 하지만 이번 SIC/XE 어셈블러에서는 체크하지 않는다.

* Compile 해야 되는 기본 소스파일 p55 Figure 2.5

3. 환경: 개별 프로젝트임!!

Linux (gcc): 반드시 gcc만을 이용해서 C언어로 프로그램 하십시오.

특히 C언어가 아닌 C++ 등 다른 언어를 사용하거나, 도스 및 윈도우에서 작성한 경우 0점 처리합니다.

참고) 컴파일 시, make 파일에 gcc –Wall 옵션을 사용하여 warning 을 철저히 확인할 것. (Warning 발생시 감점 처리함.)

4. Due Date:

4월 6일(수) 23:59시까지 제출. 늦을 경우 하루에 10%씩 감점.

5. 제출물 (아래 파일들이 모두 포함되어 있어야 함)

1) 프로그램 소스 및 헤더파일
2) Makefile
3) 프로그램 다큐멘테이션 리포트:
   이번에는 XE 소스 assemble이 주된 기능이니 만큼 이에 대한 프로그램 흐름이나 알고리즘 설명 (어떻게 구현하였는지)을 꼭 넣어준다.
4) 프로그램의 컴파일 방법 및 실행방법에 대한 간단한 내용을 적은 README파일
5) 기타 수행에 필요한 파일 (ex) opcode.txt .....)
6) 테스트 파일 (ex) 2_5.asm 등

6. 제출 방법

sp학번_proj2 이름의 디렉터리를 만들고, 여기에 위에서 설명한 모든 파일들을 넣은 후, **디렉터리를 tar로 압축하여 한 파일로 만들어** 사이버캠퍼스 과제란에 제출한다. (압축파일 내에 반드시 디렉터리가 포함되어 있어야 하며, 바이너리파일 및 코어파일을 제외할 것. 기타 불필요한 파일을 포함시키지 말 것.)

ex) sp20191234_proj2/

README → 컴파일 방법 및 실행방법에 대한 간단한 내용을 적은 파일

Document.doc →(또는 Document.docx)

20191234.c → 소스 파일이 여러 개인 경우 main 함수가 있는 파일의 이름을 학번.c 로 한다.

20191234.h → 최소 한 개 이상의 헤더 파일. 하나인 경우 학번.h

Makefile → 실행파일은 20131234.out처럼 학번.out 이름으로 고정할 것.

opcode.txt → 프로젝트#1에서 제공된 opcode 파일.

2_5.asm → 제공되는 테스트파일.

tar 명령어는 아래와 같이 사용한다.

tar 파일로 묶을 때 지난 project와 동일하게 –z 옵션을 사용하지 않는다.

tar 파일의 이름은 다음과 같이 지정한다.

sp학번_proj2.tar

ex) sp20191234_proj2.tar

제출 주소 : **사이버 캠퍼스 과제란**

파일 형식 : **[SP proj#2]_학번_이름**

(예: [SP proj#2]_20191234_홍길동)

주의사항

제출시 첨부할 파일이 잘 작성되었는지 확인하고 제출한다.

+ 제출형식(파일제목, tar file 이름 형식, 내용물)이 잘못되었을 시, 감점 10%

+ 제출 시간이 늦춰질 시, 감점

24시간(1일) 이내 10%감점

2일 이내 20%감점

3일 이내 30% 감점

4일 이내 40% 감점

5일 이내 50% 감점, 그 이상은 100% 감점

(지각 제출 시 : 김기현 조교, 명세서 하단 연락처 참고)

7. Source code 관련

Compile error

Compile error로 실행이 불가능한 경우: 숙제 전체 0점, (makefile이 없는 경우도 마찬가지임)

Segmentation fault

실행 불가 시: 0점

명령 수행 시: 그 부분점수 0점

Warning

1건당 1점 감점

Test case

2_5.asm파일과 2_5.asm을 변형한 예제 파일 수행

주석

주석이 없거나, 알아볼 수 없는 경우 감점.

타인이 알아볼 수 있는 형태로 주석을 작성할 것.

***** 모든 프로그램은 자동 검증 프로그램에 의해서 복사 검사가 됩니다. 절대로 타인의 프로그램을 참조하지 말도록 하세요. 무조건 F가 나갑니다. ******

"해당 프로젝트는 완벽한 어셈블러를 구현하는 것이 아닙니다. 채점시 기본 테스트 케이스(2_5.asm)과 기본 테스트 케이스를 변형한 테스트 케이스를 사용할 것이기 때문에 예외처리가 아닌 기능 구현에 중점을 두고 프로젝트를 수행하기 바랍니다."

8. 프로젝트에 대한 질문사항은 사이버캠퍼스 질의응답 게시판을 이용해 주시거나 조교에게 연락해 주시기 바랍니다.

김기현: zexyprofessional16@gmail.com