과목 명: 시스템프로그래밍

담당 교수 명: 박 운 상

<<Assignment 3>>

**서강대학교 컴퓨터학과**

**[20131575]**

**[유승재]**

목 차

1. 프로그램 개요 4

1.1 주소 지정 명령어 4

1.1.1 progaddr [address] 4

1.2 Linking Loader 4

1.2.1 loader [object filename1] [object filename2] […] 4

1.3 프로그램 실행 4

1.3.1 run 4

1.4 debug 명령어 5

1.4.1 bp […] 5

2. 프로그램 설명 6

2.1 프로그램 흐름도 6

3. 모듈 정의 7

3.1 모듈 이름 : progaddr\_set(char\* s, long \*a) 7

3.1.1 기능 7

3.1.2 사용 변수 7

3.2 모듈 이름: loader\_read(\_\_objcode \*\*obc, char \*filename, int fileNumber) 7

3.2.1 기능 7

3.2.2 사용 변수 7

3.3 모듈 이름: loader\_pass1(\_\_objcode \*\*obc, \_\_symtab \*\*est, long PROGADDR, int fileNumber) 7

3.3.1 기능 7

3.3.2 사용 변수 7

3.4 모듈 이름: loader\_pass2(\_\_objcode \*\*obc, \_\_symtab \*\*est, long PROGADDR, int fileNumber, unsigned char\* m) 7

3.4.1 기능 7

3.4.2 사용 변수 7

3.5 모듈 이름: loader\_free(\_\_objcode \*\*obc, int fileNumber) 8

3.5.1 기능 8

3.5.2 사용 변수 8

3.6 모듈 이름: loader\_loadmapPrint(\_\_symtab \*\*es) 8

3.6.1 기능 8

3.6.2 사용 변수 8

3.7 모듈 이름: loader\_loadmapPrintCmp(const void \*arg1, const void \*arg2) 8

3.7.1 기능 8

3.7.2 사용 변수 8

3.8 모듈 이름: run\_process(unsigned char \*m, \_\_breakpoint \*b, \_\_opcode \*\*o, long prog, long \*exec, long len, int flag) 8

3.8.1 기능 8

3.8.2 사용 변수 8

3.9 모듈 이름: run\_register\_reset(void) 8

3.9.1 기능 8

3.9.2 사용 변수 9

3.10 모듈 이름: run\_register\_input(char\* rn, unsigned int input) 9

3.10.1 기능 9

3.10.2 사용 변수 9

3.11 모듈 이름: run\_register\_value(char \*rn) 9

3.11.1 기능 9

3.11.2 사용 변수 9

3.12 모듈 이름: run\_register\_store(char\* rn, unsigned char\* m, int adr) 9

3.12.1 기능 9

3.12.2 사용 변수 9

3.13 모듈 이름: run\_register\_print(void) 9

3.13.1 기능 9

3.13.2 사용 변수 9

3.14 모듈 이름: bp\_input(\_\_breakpoint\* b, unsigned long adr) 9

3.14.1 기능 9

3.14.2 사용 변수 9

3.15 모듈 이름: bp\_print(\_\_breakpoint \*b) 10

3.15.1 기능 10

3.15.2 사용 변수 10

3.16 모듈 이름: bp\_clear(\_\_breakpoint \*b) 10

3.16.1 기능 10

3.16.2 사용 변수 10

3.17 모듈 이름: memory\_find(unsigned char \*m, long address) 10

3.17.1 기능 10

3.17.2 사용 변수 10

3.18 모듈 이름: opcode\_operation(\_\_opcode \*\*o, int code) 10

3.18.1 기능 10

3.18.2 사용 변수 10

4. 전역 변수 정의 10

4.1 unsigned char memory[MEMORYSIZE] 10

4.2 \_\_DIRECTIVES directive[] 10

4.3 \_\_REGISTERS registers[REGNUM] 10

5. 코드 11

5.1 20111633.h 11

5.2 20111633.c 18

# 프로그램 개요

본 프로젝트 #2 에서 구현한 셀(shell)에 linking과 loading 기능을 추가하는 프로그램으로, 프로젝트 #2 에서 구현된 assemble 명령을 통해서 생성된 object 파일을 link시켜 메모리에 올리는 일을 수행한다. 주소 지정 명령어, Linking Loader, 프로그램 실행 명령어, debug 명령어 등을 구현해야 하는데 이를 위해 필요한 자료구조 및 알고리즘을 구상하여 전체 프로그램이 설계되어 있다.

이 프로그램을 실행시키면 “sicsim>”(unix shell과 유사한 입력 프롬프트상태)가 된다. 이 상태에서 아래에 있는 명령어들을 입력할 때, 그에 해당되는 기능을 수행하여야 한다.

.

## 주소 지정 명령어

### progaddr [address]

- loader 또는 run 명령어를 수행할 때 시작하는 주소를 지정한다.

- sicsim이 시작되면 default로 progaddr는 0x00 주소로 지정된다..

## Linking Loader

### loader [object filename1] [object filename2] […]

- - filename1, filename2, … 에 해당하는 object 파일을 읽어서 linking 작업을 수행 후, 가상 메모리(1M)에 그 결과를 기록한다.

- Loader 실행이 성공적이면, load map을 화면에 출력한다.

- 에러가 존재할 경우, 에러 내용이 화면에 출력된다.

## 프로그램 실행

### run

- loader 명령어의 수행으로 메모리에 load된 프로그램을 실행한다.

( 프로그램의 구성에 I/O 명령어는 제외한다. )

- progaddr 명령어로 지정한 주소부터 실행된다.

- 실행 결과로써 register 상태를 화면에 출력한다.

출력되는 register는 A, X, L, PC, B, S, T 이다.

- Breakpoint까지 실행되고 Breakpoint가 없으면 프로그램 끝까지 실행된다.

( 명령어 bp 참조 )

## debug 명령어

### bp […]

* […] = [address]
  + sicsim에 breakpoint를 지정한다.
  + Breakpoint는 bp 명령어를 통해서 무제한으로 지정할 수 있다.
  + run을 수행하면 breakpoint까지 프로그램이 실행되고 프로그램이 정지한다. 다음 번 run의 실행은 정지된 breakpoint부터 시작하고 그 다음 breakpoint까지 진행된다. Breakpoint가 없으면 프로그램 끝까지 실행된다.
* […] = cleaer
  + sicsim에 존재하는 breakpoint를 전부 삭제한다.
* […] = 입력 값 없음
  + sicsim에 존재하는 breakpoint를 전부 화면에 출력한다.

# 프로그램 설명

## 프로그램 흐름도



그림 1> 프로그램 흐름도

프로그램 실행 시 받은 명령어를 통해 적절한 행위를 수행한다. 다만, 프로그램이 중간에 오류가 있다면, 오류가 있는 이유를 출력하고 프로그램이 수월하게 지속될 수 있도록 수행을 중단한다.

# 모듈 정의

## 모듈 이름 : progaddr\_set(char\* s, long \*a)

### 기능

입력 받은 문자열 s을 16진수로 계산하여 a에 돌려준다. 해당 값이 잘 나왔다면 1을, 변환 과정에 오류가 있다면 0을 반환한다.

### 사용 변수

char \*inputCheck – s의 값에 입력된 값의 유효성을 확인하는 문자형 변수

long calAdr– s를 통해 계산된 값이 저장되는 정수형 변수

## 모듈 이름: loader\_read(\_\_objcode \*\*obc, char \*filename, int fileNumber)

### 기능

입력 받은 문자열 filename의 이름을 갖는 flieNumber개수만큼 소스 파일들을 읽어 오브젝트 구조체 obc에 저장한다. 만약 읽는 파일에 문제가 있어 함수를 중단할 경우 0을, 정상적으로 실행되어 종료되면 1을 반환한다.

### 사용 변수

FILE \*\*fp – 파일 스트림

\_\_objcode \*n – obc를 이동시킬 objcode형 변수

char \*\*filenameToken – filename에 저장된 파일 이름들을 token하기 위한 문자형 변수

char \*inputCheck – 파일명 확인을 위한 문자형 변수

int line – 현재 라인을 계산할 정수형 변수

## 모듈 이름: loader\_pass1(\_\_objcode \*\*obc, \_\_symtab \*\*est, long PROGADDR, int fileNumber)

### 기능

loader\_read을 통해 저장된 obj코드 obc를 한 줄씩 파악하고, symbol이 나온다면 est에 symbol table을 저장한다. CSADDR와 CSLTH를 구한다. 만약 obc코드에 문제가 있어 함수를 중단할 경우 0을, 정상적으로 실행되어 종료되면 사용된 총 사용된 LOC의 값을 반환한다.

### 사용 변수

long CSADDR – CSADDR을 계산하기 위한 정수형 변수

long CSLTH – CSLTH을 계산하기 위한 정수형 변수

\_\_objcode \*n – obc을 이동시킬 objcode형 변수

int i – loop counter

## 모듈 이름: loader\_pass2(\_\_objcode \*\*obc, \_\_symtab \*\*est, long PROGADDR, int fileNumber, unsigned char\* m)

### 기능

loader\_pass1을 통해 처리된 obj코드 obc를 메모리 m에 올린다. 만약 symbol이 나온다면, est table이 담긴 est에서 symbol에 맞는 값을 찾는다. 만약 asm코드에 문제가 있어 함수를 중단할 경우 0을, 정상적으로 실행되어 종료되면 1을 반환한다.

### 사용 변수

long CSADDR – CSADDR을 계산하기 위한 정수형 변수

long EXECADDR – EXECADDR을 계산하기 위한 정수형 변수

long CSLTH – CSLTH을 계산하기 위한 정수형 변수

\_\_objcode \*n – obc을 이동시킬 objcode형 변수

int i – loop counter

## 모듈 이름: loader\_free(\_\_objcode \*\*obc, int fileNumber)

### 기능

\_\_objcode 구조체에 할당된 메모리를 반환하는 함수이다.

### 사용 변수

int i – loop counter

## 모듈 이름: loader\_loadmapPrint(\_\_symtab \*\*es)

### 기능

loader\_pass2를 통해 처리된 obj코드를 통해 생성된 est table es의 값을 출력한다. 함수 또는 파일에 문제가 있어 함수를 중단할 경우 0을, 정상적으로 실행되어 종료되면 프로그램의 총 길이를 반환한다.

### 사용 변수

\_\_symtab \*n - es를 이동시킬 symtab형 변수

\_\_symtab \*sortSm - es를 내림차순으로 정렬하여 저장할 symtab형 변수

int numberSm - 정렬을 시행해야 할 sm내의 symbol의 총 개수

int totalLength - 프로그램의 총 길이를 저장할 정수형 변수

int i - loop counter

int k - loop counter

## 모듈 이름: loader\_loadmapPrintCmp(const void \*arg1, const void \*arg2)

### 기능

loader\_loadmapPrint 에서 quick sort를 하기 위해 비교 방법을 정의한 함수이다.

### 사용 변수

\_\_symtab \*sm1 – arg1을 저장할 symtab형 변수

\_\_symtab \*sm2 – arg2을 저장할 symtab형 변수

## 모듈 이름: run\_process(unsigned char \*m, \_\_breakpoint \*b, \_\_opcode \*\*o, long prog, long \*exec, long len, int flag)

### 기능

메모리 m에 올려진 프로그램을 실행한다. 프로그램이 올라간 메모리의 시작은 prog부터 prog+len까지 이다. m에 올려진 프로그램을 opcode o를 사용하여 실행한다. 이 때 breakpoint인 b가 있다면 해당 지점에서 정지하여준다. bp를 사용하였다면 -1을, bp를 사용하지 않았다면 1을 반환한다.

### 사용 변수

없음

## 모듈 이름: run\_register\_reset(void)

### 기능

register의 값을 0으로 초기화 하는 함수이다.

### 사용 변수

int i - loop counter

## 모듈 이름: run\_register\_input(char\* rn, unsigned int input)

### 기능

register의 이름 rn에 맞게 그 위치에 input값을 저장하는 함수이다.

### 사용 변수

int i - loop counter

## 모듈 이름: run\_register\_value(char \*rn)

### 기능

register의 이름 rn에 맞게 그 위치에 input값을 리턴하는 함수이다..

### 사용 변수

int i – loop counter

## 모듈 이름: run\_register\_store(char\* rn, unsigned char\* m, int adr)

### 기능

register의 이름 rn에 있는 값을 메모리 m의 주소 adr부터 adr+2까지 저장하는 함수이다.

### 사용 변수

int i – loop counter

## 모듈 이름: run\_register\_print(void)

### 기능

register의 값을 출력 형식에 맞게 출력하는 함수이다.

### 사용 변수

char printset[][3]– 출력을 할 레지스터와 그 순서를 저장한 문자형 변수

int printsetlen – printset의 길이를 저장한 정수형 변수

int i - loop counter

int j - loop counter

## 모듈 이름: bp\_input(\_\_breakpoint\* b, unsigned long adr)

### 기능

breakpoint형 변수 b에 break할 adr를 저장하는 함수이다. 입력 시, adr의 값에 따라 정렬하여 넣도록 한다.

### 사용 변수

\_\_breakpoint \*now – 현재 위치의 값을 확인하기 위한 breakpoint형 변수

\_\_breakpoint \*nxt – 현재 바로 다음 값을 확인하기 위한 breakpoint형 변수

\_\_breakpoint \*i – adr이 저장된 입력할 breakpoint형 변수

## 모듈 이름: bp\_print(\_\_breakpoint \*b)

### 기능

breakpoint형 변수 b에 있는 값을 출력하는 함수이다.

### 사용 변수

\_\_breakpoint \*n – b를 이동시킬 breakpoint형 변수

## 모듈 이름: bp\_clear(\_\_breakpoint \*b)

### 기능

\_\_breakpoint 구조체에 할당된 메모리를 반환하는 함수이다.

### 사용 변수

\_\_breakpoint \*n – b를 이동시킬 breakpoint형 변수

\_\_breakpoint \*f – b를 할당 해제할 breakpoint형 변수

## 모듈 이름: memory\_find(unsigned char \*m, long address)

### 기능

memory m에서 address위치의 값을 찾아주는 함수이다.

### 사용 변수

없음

## 모듈 이름: opcode\_operation(\_\_opcode \*\*o, int code)

### 기능

opcode o에서 code에 맞는 opcode를 찾아 반환하는 함수이다.

### 사용 변수

\_\_opcode \*n – o을 이동시킬 opcode형 변수

int i - loop counter

# 전역 변수 정의

## unsigned char memory[MEMORYSIZE]

SIC/XE 머신을 구현하기 위한 메모리공간. (1Mbyte)

## \_\_DIRECTIVES directive[]

SIC/XE 머신의 directive 명령어를 저장한 공간. (RESB~RESQ / BYTE~QWORD)

## \_\_REGISTERS registers[REGNUM]

SIC/XE 머신의 Register의 번호에 맞게 Mnemonic을 저장한 공간. (A,X…,SW)

# 코드

코드는 깃헙 페이지 <https://github.com/sogangCSmaster/sogang_sp_project3> 에 전체 공개로 해 두었다.