**과목명: 시스템프로그래밍**

**02 분반**

**<<Project #3>>**

**서강대학교 컴퓨터공학과**

**20181614**

**김주연**

목 차

1. **프로그램 개요**
   1. 프로젝트 문제
   2. 프로젝트 목표
2. **프로그램 설명**
   1. linking loader 흐름도
   2. 프로그램 설명
3. **모듈 정의**
   1. 정의한 모듈에 대해 간략한 설명
4. **전역 변수 정의**
5. **코드 설명**
6. **프로그램 개요**

1.1 프로젝트 문제

프로젝트 1, 2 에서 구현한 셀(shell)에 linking과 loading 기능을 추가하는 프로그램이다. 프로젝트 2 에서 구현된 assemble 명령을 통해서 생성된 object 파일을 link시켜 메모 리에 올리는 일을 수행한다. 주소 지정 명령어, Linking Loader, 프로그램 실행 명령어, debug 명령어 등을 구현해야 하는데 이를 위해 필요한 자료구조 및 알고리즘을 구상하여 전체 프로그램을 설계한다.

1.2 프로젝트 목표

이미 제출한 프로젝트 2에 아래의 기능들을 추가한다.

* 주소 지정 명령어 (progaddr)
* Linking Loader (loader)
* 프로그램 실행 (run)
* debug 명령어 (bp)

1. **프로그램 설명**
   1. linking loader 흐름도

이는 프로젝트 3에서 중점이 되는 linking loader 기능의 흐름도이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. 프로그램 설명

progaddr 기능으로 loader 또는 run 명령어를 수행할 때 시작할 주소를 설정할 수 있다. 이후 작성한 loader 기능으로 입력 받은 이름에 해당하는 오브젝트 파일을 읽어 와 pass 1과 pass 2의 과정을 거쳐 프로그램을 가상 메모리(1M)에 올리는 과정을 수행한다.

pass 1에서는 H record와 D record에서 해당 control section의 이름과 주소, defined external symbol의 이름과 주소를 읽어 와서 CSADDR값을 고려한 실제 주소 값으로 ESTAB(external symbol table)에 저장한다. E record를 만나면 종료한다. 모든 오브젝트 파일에 대해 해당 기능을 수행하고 나면 로드맵(ESTAB 내용)을 출력한다.

pass2에서는 R record의 reference number와 변수 이름을 array 형식으로 저장한다. 또한 T record의 내용을 EXECADDR가 가리키는 가상 메모리에 저장한다. 이후 pass 1에서 생성한 ESTAB의 내용을 이용하여 R record의 변수들에 대한 relocation 작업을 수행한다. pass 1과 pass 2 과정을 통해 프로그램이 가상 메모리의 알맞은 위치에 로딩된다. 또한 여러 오브젝트 파일이 들어온 경우 링킹 작업 또한 수행되었다.

run 기능은 위의 loading 과정 이후 실제로 해당 프로그램을 실행하여 레지스터와 메모리 값의 변화를 볼 수 있도록 하였다. copy.obj 파일에 대해 수행되도록 하였으며, bp 기능을 통해 break point를 지정하여 PC가 break point값을 가리키는 경우 프로그램이 해당 지점까지만 수행될 수 있도록 하였다.

progaddr와 linking loader 기능은 loader.c에 작성하였으며, run과 bp 기능은 run.c에 작성하였다.

1. **모듈 정의**
   1. 정의한 모듈에 대해 간략한 설명 (프로젝트 1, 2 내용 제외)

20181614.c 파일

* main() :

parameter : void

return : 프로그램 종료 시 0을 return한다.

기능 : 프로젝트 1, 2에서 구현한 기능에 프로젝트 3에서 필요한 기능을 추가하였다. 초기 progaddr 값을 0x00으로 초기화하였으며, break point를 저장할 BLIST\* 변수 bHead를 초기화하였다. 또한 switch 문에서 이번에 추가된 progaddr, loader, run, bp 기능을 선택할 수 있도록 추가하였다. 그때 20181614.h 헤더 파일에 존재하는 ENUM list에 추가된 PROGADDR, LOADER, RUN, BP를 switch문에 이용하였다.

shell.c 파일

* hFunc() :

parameter : void

return : void

기능 : 이 프로그램에서 수행할 수 있는 기능들의 리스트를 출력한다. 이번에 추가된 progaddr, loader, run, bp 기능을 리스트에 추가하였다.

loader.c 파일

* pFunc() :

parameter : 사용자가 입력한 시작 주소 문자열

return : void

기능: 사용자가 입력한 프로그램의 시작 주소를 문자열로 받아서 그것을 16진수 숫자로 변환한 뒤 PROGADDR값을 나타내는 전역변수 prog\_addr에 저장하였다.

* lpass1() :

parameter : 사용자가 입력한 object file name string

return : 오류가 있다면 -1, 아니라면 0을 return한다.

기능: object file(s)을 입력으로 받아서, 각각의 H record에서 프로그램 이름을 받아 CSADDR값을 고려한 실제 주소와 함께 ESTAB에 저장한다. 또한 D record가 있다면 D record에 있는 모든 external symbol들을 CSADDR값을 고려한 실제 주소와 함께 ESTAB에 저장한다. 이후 H record에서 읽었던 프로그램의 길이를 CSADDR에 저장하고, 여러 개의 파일이 입력으로 들어왔다면 다음 파일에 대해 같은 작업을 수행한다.

* lpass2() :

parameter : void

return : 0

기능: pass 1에서 받았던 object file(s)을 열어서, R record가 있다면 reference number와 함께 array에 referred external symbol들을 저장한다. 이후 T record를 읽어서 해당 프로그램이 기록될 위치에 맞추어 가상 메모리에 object code를 올린다. 모든 Text record를 읽은 뒤, M record가 있다면 위의 reference array와 pass 1에서 기록한 ESTAB을 사용하여 해당 external symbol의 실제 주소를 받아 와 relocation을 수행한다.

* esnodeConnect () :

parameter : ESTAB head, node

return : void

기능 : program name 또는 external symbol에 대해 ESTAB hash table에 저장하는 역할을 수행한다.

* StrtoHex() :

parameter : char

return : 16진수로 변환한 int 값

기능 : char형 변수를 그 값에 해당하는 16진수 숫자로 변환하여 return한다.

* findMod() :

parameter : object file 번호, reference number

return : 해당 external symbol의 실제 주소

기능 : reference array는 object file마다 reference number와 그에 해당하는 external symbol을 저장하고 있으므로, 현재 object file의 번호와 reference number를 받아서 refer된 external symbol의 이름을 찾을 수 있다. 이후 해당 이름을 ESTAB에서 검색하여 그 external symbol의 실제 주소를 찾아 return한다.

run.c 파일

* rFunc() :

parameter : void

return : 0

기능 : loader 기능을 사용하여 가상 메모리에 올렸던 object file의 내용을 수행한다. PC가 현재 메모리의 위치를 가리키고 있으며, PC의 주솟값이 프로그램의 길이를 넘어가거나 시작 주소보다 작아지기 전까지 계속해서 프로그램이 수행되게 된다. PC가 가리키는 메모리에서 object code를 읽어 와 그를 opcode와 n, i,x, b, p, e, disp로 다시 분리하고, format마다 opcode에 맞는 기능을 수행하도록 하였다. 만약 PC가 break point와 같다면 해당 지점까지만 프로그램을 수행하고, 다시 run을 하면 그 다음 지점부터 프로그램이 다시 수행되도록 하였다.

* findFormat () :

parameter : opcode 값

return : 해당 mnemonic의 format

기능 : opcode 값을 입력으로 받아 op table에서 해당 mnemonic을 검색하여 format을 return한다.

* returnReg() :

parameter : register 번호

return : 해당 register 정보를 저장하는 변수의 주솟값

기능 : format 2에서 register 연산을 수행하기 위해 작성한 함수이다. register 번호를 입력으로 받아 해당 register 정보를 저장하고 있는 전역변수의 주솟값을 return한다.

* format2() :

parameter : opcode, register 번호

return : void

기능 : format 2에 해당하는 mnemonic들의 기능을 담고 있는 함수이다. opcode와 register 번호를 입력으로 받아 opcode에 따라 다른 연산을 수행한다. copy.obj 연산 수행을 위한 mnemonic들을 포함하고 있다.

* format3() :

parameter : opcode, n, i, x, b, p, e, disp

return : void

기능 : format 3, 4에 해당하는 mnemonic들의 기능을 담고 있는 함수이다. opcode와 n, i, x, b, p, e, disp를 입력으로 받아 각각을 분리하여 저장하고, n, i, x, b, p, e 값에 따라서 PC relative, Base relative, Indexing 등과 Immediate addressing, Indirect addressing, Simple addressing 등 종류에 따라 다시 연산을 수행하여 Target address를 찾는다. 이후 opcode와 target address에 따라 다른 연산을 수행한다. copy.obj 연산 수행을 위한 mnemonic들을 포함하고 있다.

* bFunc() :

parameter : break point 문자열

return : 16진수로 변환한 int값

기능 : 사용자가 입력한 break point를 문자열로 받아서, 만약 아무 입력이 없었다면 bp의 기능인 ‘break point 목록 출력’을 수행하고 만약 입력이 clear였다면 bp clear의 기능은 ‘모든 break point 삭제’ 기능을 수행한다. 이외의 경우 break point를 16진수로 변환하여 break point를 linked list 형태로 저장하고 있는 struct BLIST에 연결하여 준다.

1. **전역 변수 정의**

20181614.h 파일

* enum commEnum

: 프로젝트 1, 2에서 지정한 keyword에 이번에 추가된 PROGADDR, LOADER, RUN, BP를 추가하였다.

* struct estab, ESTAB \*\*eHead

: linking & loading 과정 중에 external symbol을 저장하기 위한 table이다. external symbol의 이름과 주소를 저장하고 있다. 파일마다 다른 ESTAB을 가져야 하므로 table은 이중 포인터 eHead로 선언하였다.

* prog\_addr, cs\_addr, exec\_addr, end

: 각각 PROGADDR, CSADDR, EXECADDR의 역할을 하는 전역 변수이며, end는 loader 과정 이후 프로그램의 길이를 저장하고 있다.

* struct bpList, BLIST \*pHead

: break point를 linked list 형태로 저장한다.

* A, X, L, B, S, T, F, PC, SW

: run을 수행할 때 register의 역할을 하는 전역 변수이다. loader 과정 이후 L은 프로그램의 길이를 저장하고 있다.

* CC

: opcode에 따른 연산 수행 중, 대소 비교와 관련하여 값을 저장하는 변수이다.

loader.c 파일

* objcnt

: 입력으로 들어 온 object file의 개수를 저장하는 변수이다.

* str

: object file을 한 줄 씩 읽어올 때 사용하는 변수이다.

* ESARR

: pass 2에서 reference number에 따른 external symbol을 저장하기 위한 array이다. reference number가 index, external symbol이 내용이 되며, object file마다 다른 array를 사용해야 하므로 char 형 3중 포인터를 사용하여 선언하였다.

* p

: object file을 열 때 사용하는 file pointer이다.

* ePoint

: ESTAB에서 symbol 검색 시 사용하는 변수이다.

1. **코드 설명**

코드 설명은 위의 모듈 설명에 더해 주석으로 하였다.

* loader.c 파일

스크린샷, 전화, 노트북이(가) 표시된 사진

자동 생성된 설명

스크린샷, 전화, 앉아있는, 휴대폰이(가) 표시된 사진

자동 생성된 설명

전화, 노트북, 앉아있는, 휴대폰이(가) 표시된 사진

자동 생성된 설명

앉아있는, 노트북, 전화, 검은색이(가) 표시된 사진

자동 생성된 설명

스크린샷, 앉아있는, 전화, 테이블이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

그리기이(가) 표시된 사진

자동 생성된 설명

* run.c 파일

스크린샷, 앉아있는, 전화, 모니터이(가) 표시된 사진

자동 생성된 설명

스크린샷, 노트북, 전화이(가) 표시된 사진

자동 생성된 설명

스크린샷, 전화이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

텔레비전, 모니터, 화면, 제어이(가) 표시된 사진

자동 생성된 설명

스크린샷, 노트북, 앉아있는, 컴퓨터이(가) 표시된 사진

자동 생성된 설명