과목 명: 시스템프로그래밍

담당 교수 명: 소 정 민

<<Assignment 1>>

**서강대학교 컴퓨터공학과**

**[20161631]**

**[임동진]**

목 차

1. 프로그램 개요 3

2. 프로그램 설명 3

2.1 프로그램 흐름도 3

3. 모듈 정의 5

3.1 모듈 이름 : shell\_status execute\_progaddr(Command \*user\_command, State \*state\_store) 5

3.1.1 기능 5

3.1.2 사용 변수 5

3.2 모듈 이름: display\_instructions() 5

3.2.1 기능 5

3.2.2 사용 변수 5

3.3 모듈이름: get\_data() 5

3.3.1 기능 5

3.3.2 사용변수 5

3.4 모듈이름: display\_report() 5

3.4.1 기능 5

3.4.2 사용변수 5

3.5 모듈이름: continue\_function() 5

3.5.1 기능 5

3.5.2 사용변수 5

3.6 모듈이름: Hello world! 6

3.7 Asdasdsaas 6

3.7.1 ssssssssssadsasd 6

3.8 sadasassadasda 6

ss 6

4. 전역 변수 정의 6

4.1 long income[100] 6

4.2 int month[100], day[100], year[100] 6

4.3 int ctr 6

4.4 int cont 6

5. 코드 6

# 프로그램 개요

SIC/XE 의 Linking Loader를 구현한다. 여러 object file을 linking하여 메모리에 load하는 작업을 수행한다. 또한 load 된 프로그램을 실행하고, 브레이크 포인트를 통해 디버깅할 수 있는 디버깅 기능까지 구현되어있다.

# 프로그램 설명

## 프로그램 흐름도

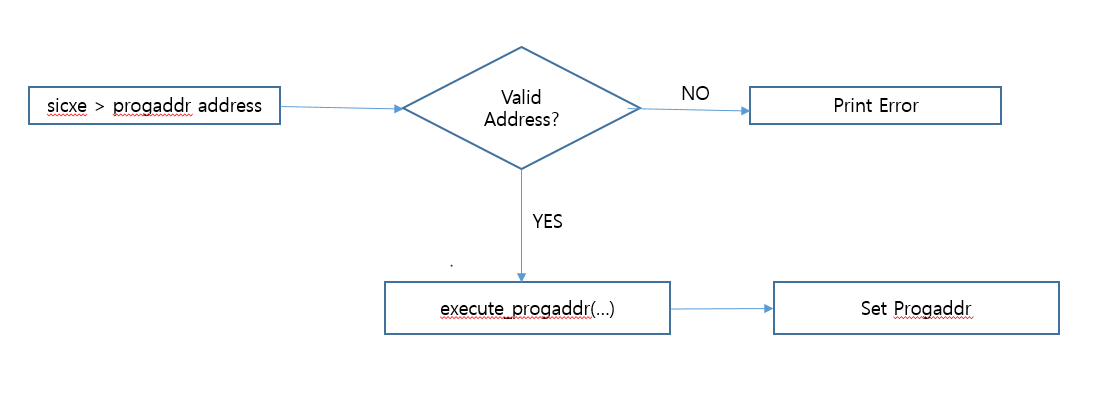


그림 1> progaddr 명령 흐름도

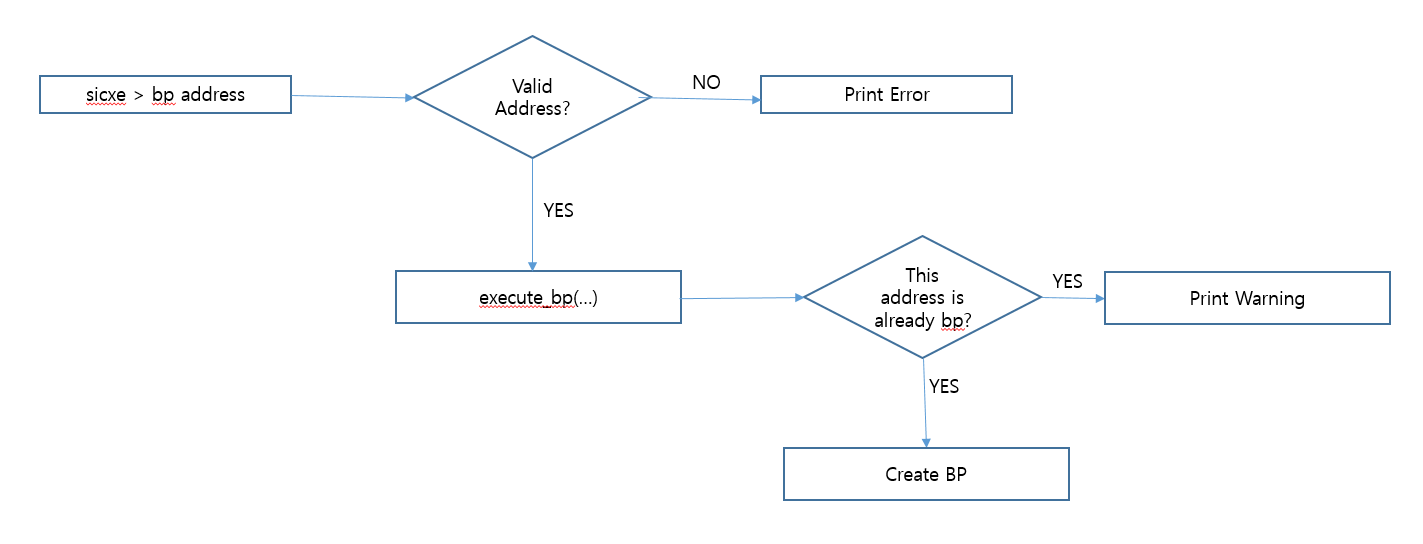


그림 2> bp [address] 명령 흐름도



그림 3> bp 명령 흐름도

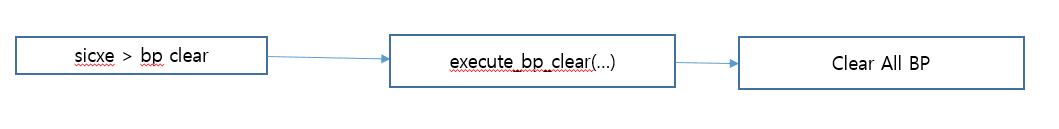


그림 4> bp clear 명령 흐름도

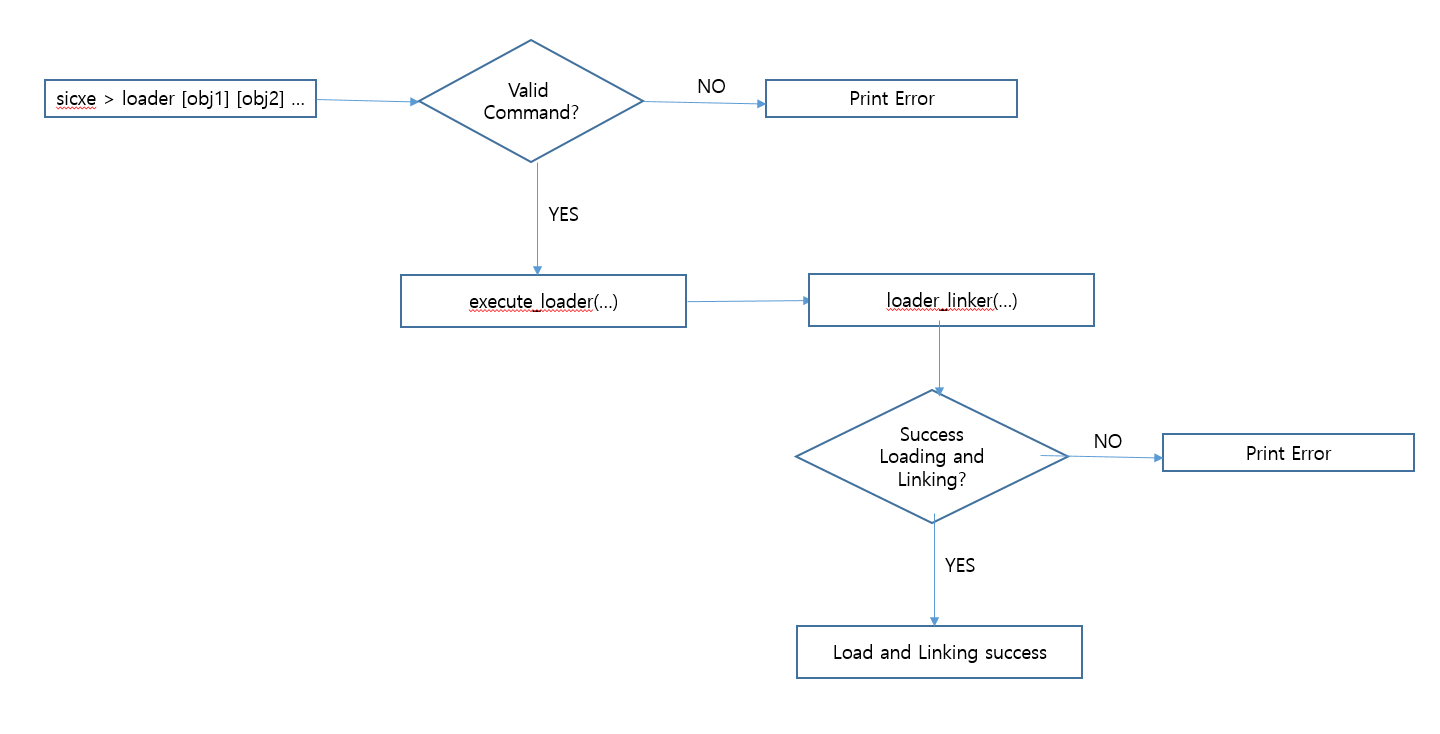


그림 5> loader [obj1] [obj2] … 명령 흐름도

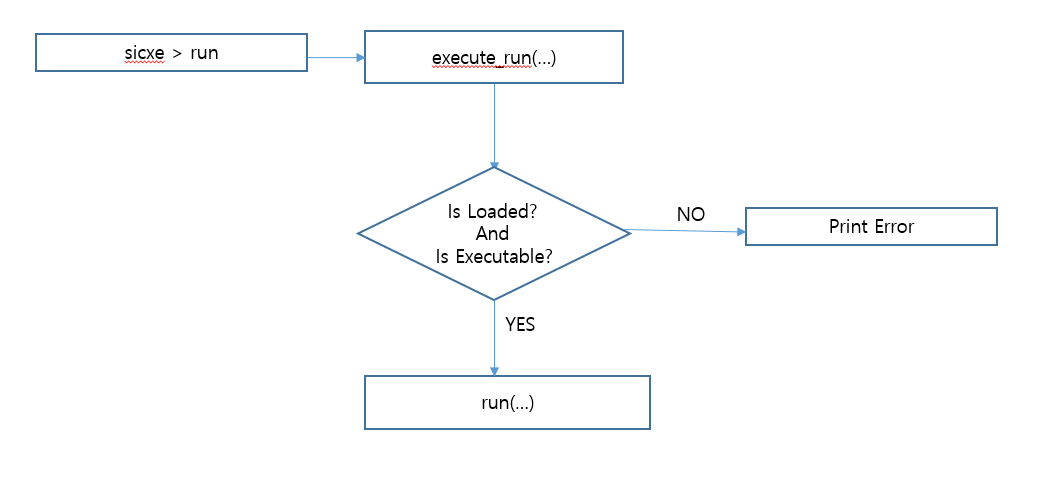


그림 6> run 명령 흐름도

# 모듈 정의

## 모듈 이름 : execute\_progaddr(Command \*user\_command, State \*state\_store)

### 기능

progaddr [address] 명령을 수행한다. 프로그램이 load 될 시작 주소를 지정하는 명령이다. ..

### 사용 변수

|  |  |
| --- | --- |
| Command \*user\_command | .사용자가 입력한 명령에 대한 정보를 저장한다. |
| State \*state\_store | 현재 프로그램의 모든 상태정보를 저장한다. |
| int addr | Progaddr [address]에서 address로 들어온 값을 저장한다. |

## 모듈 이름: execute\_run(State \*state\_store)

### 기능

현재 프로그램이 run 될수 있는지 확인하고, run 할수있다면 run(…)함수를 호출하고, 아니라면 에러를 출력해준다.. 실질적인 run 명령 수행은 run(…) 함수에서 수행된다.

run(..) 함수 수행 성공 실패 여부에 따라서 적절한 에러 핸들링을 진행한다.

### 사용 변수

|  |  |
| --- | --- |
| bool status; | run(..)함수 수행의 성공 실패 여부를 저장한다.. |
| State \*state\_store | 현재 프로그램의 모든 상태정보를 저장한다. |

## 모듈이름execute\_bp(Command \*user\_command, State \*state\_store)

### 기능

bp [address] 명령을 수행한다. 브레이크 포인트를 설정한다. 이미 존재하는 bp라면 경고를 출력한다.

### 사용변수

|  |  |
| --- | --- |
| Command \*user\_command | .사용자가 입력한 명령에 대한 정보를 저장한다. |
| State \*state\_store | 현재 프로그램의 모든 상태정보를 저장한다. |
| int addr | Progaddr [address]에서 address로 들어온 값을 저장한다. |

## 모듈이름: execute\_bp\_clear(State \*state\_store)

### 기능

bp clear 명령을 수행한다. 모든 브레이크 포인트를 삭제한다..

### 사용변수

|  |  |
| --- | --- |
| State \*state\_store | 현재 프로그램의 모든 상태정보를 저장한다. |
| int cnt | 반복문 수행을 위한 보조 변수 |

.

## 모듈이름: execute\_loader(Command \*user\_command, State \*state\_store)

### 기능

loader\_linker(…) 함수를 호출한다. 실질적인 loader [obj1] [obj2] … 명령 수행은 loader\_linker(…) 함수에서 수행된다. loader\_linker(..) 함수 수행 성공 실패 여부에 따라서 적절한 에러 핸들링을 진행한다

### 사용변수

|  |  |
| --- | --- |
| Command \*user\_command | .사용자가 입력한 명령에 대한 정보를 저장한다. |
| State \*state\_store | 현재 프로그램의 모든 상태정보를 저장한다. |
| bool status | loader\_linker(…) 함수 수행의 성공 실패 여부를 저장한다. |

## 모듈이름: execute\_bp\_list(State\* state\_store)

### 기능

bp 들의 목록을 출력하는 명령인 bp 명령을 수행한다.

### 사용변수

|  |  |
| --- | --- |
| State \*state\_store | 현재 프로그램의 모든 상태정보를 저장한다. |
| int cnt | 반복문 수행을 위한 보조 변수 |

## 모듈이름: validate\_progaddr\_parameters(Command \*user\_command)

#### 기능

progaddr [address] 명령어의 파라미터를 검증한다. 검증 결과에 따라서 VALID\_PARAMETERS 또는 INVALID\_PARAMETERS를 반환하여준다.

#### 사용변수

|  |  |
| --- | --- |
| Command \*user\_command | .사용자가 입력한 명령에 대한 정보를 저장한다. |

## 모듈이름: validate\_bp\_parameters(Command\* user\_command

### 기능

bp [address] 명령어의 파라미터를 검증한다. 검증 결과에 따라서 VALID\_PARAMETERS 또는 INVALID\_PARAMETERS를 반환하여준다.

### 사용변수

|  |  |
| --- | --- |
| Command \*user\_command | .사용자가 입력한 명령에 대한 정보를 저장한다. |

## 모듈이름: validate\_bp\_clear\_parameters(Command\* user\_command)

### 기능

bp clear 명령어의 파라미터를 검증한다. 검증 결과에 따라서 VALID\_PARAMETERS 또는 INVALID\_PARAMETERS를 리턴하여준다

### 사용변수

|  |  |
| --- | --- |
| Command \*user\_command | .사용자가 입력한 명령에 대한 정보를 저장한다. |

## 모듈이름: validate\_loader\_parameters(Command\* user\_command)

### 기능

loader [obj1] [obj2] … 명령어의 파라미터를 검증한다. 검증 결과에 따라서 VALID\_PARAMETERS 또는 INVALID\_PARAMETERS를 리턴하여준다

### 사용변수

|  |  |
| --- | --- |
| Command \*user\_command | .사용자가 입력한 명령에 대한 정보를 저장한다. |

## 모듈이름: op\_format\_by\_op\_num(int op\_num)

### 기능

Opcode 에 따라서 대응 되는 format(1 또는 2 또는 3/4)을 리턴한다.

### 사용변수

|  |  |
| --- | --- |
| int op\_num | Opcode 정보 |
| bool format3\_4[260] | Format3\_4에 해당하는 opcode 정보를 true로 하여 저장함 |
| bool format2[260] | Format2에 해당하는 opcode 정보를 true로 하여 저장함 |

## 모듈이름: registers 구조체

### 기능

레지스터 정보를 저장한다..

### 정의 내용

|  |
| --- |
| typedef struct registers {  uint32\_t A;  uint32\_t L;  uint32\_t X;  uint32\_t PC;  uint32\_t B;  uint32\_t S;  uint32\_t T;  uint32\_t SW;  }Registers; |

## 모듈이름: load\_info\_node구조체

### 기능

load info\_list 구조체의 node 구조체 역할을 한다.

### 정의 내용

|  |
| --- |
| typedef struct load\_info\_node {  enum load\_info\_type type;  char name[15];  int addr;  int length;  }LoadInfoNode; |

## 모듈이름: load\_info\_list구조체

### 기능

loader [obj1] [obj2] …명령의 로딩된 결과를 저장한다.

### 정의 내용

|  |
| --- |
| typedef struct load\_info\_list {  LoadInfoNode list[1001];  int count;  } LoadInfoList;; |

## 모듈이름: debugger구조체

### 기능

load, run, bp, 레지스터 관리 등의 역활을함. 즉 해당 명령들의 수행을 위해 필요한 정보들을 통합해서 관리한다.

### 정의 내용

|  |
| --- |
| typedef struct debugger {  uint32\_t start\_address;  int end\_address;  bool break\_points[MAX\_BP\_NUM];  int bp\_count;  Registers\* registers;  int run\_count;.  int previous\_bp;  char\* filenames[3];  int file\_count;  SymbolTable\* estab;  LoadInfoList\* load\_infos;  uint32\_t total\_length;  bool is\_running;  bool is\_loaded;  } Debugger; |

## 모듈이름: instruction구조체

### 기능

Instruction과 관련한 정보를 저장한다..

### 정의 내용

|  |
| --- |
| typedef struct instruction{  bool extend;  unsigned char opcode;  struct {  union{  struct{  uint16\_t r2 : 4;  uint16\_t r1 : 4;  uint16\_t opcode : 8;  } p2;  struct{  uint32\_t address: 12;  uint32\_t e : 1;  uint32\_t p : 1;  uint32\_t b : 1;  uint32\_t x : 1;  uint32\_t i : 1;  uint32\_t n : 1;  uint32\_t opcode : 6;  } p3;  struct{  uint32\_t address: 20;  uint32\_t e : 1;  uint32\_t p : 1;  uint32\_t b : 1;  uint32\_t x : 1;  uint32\_t i : 1;  uint32\_t n : 1;  uint32\_t opcode : 6;  } p4;  uint32\_t val;  } param;  bool extend;  } param;  } Instruction; |

## 모듈이름: load\_info\_type ENUM

### 기능

Load 된 정보의 유형을 enum으로 설정

### 정의 내용

|  |
| --- |
| enum load\_info\_type {  INFO\_TYPE\_CONTROL\_SECTION,  INFO\_TYPE\_SYMBOL  } LoadInfoType; |

## 모듈이름: Operator ENUM

### 기능

Operator 이름과 opcode를 매핑한 enum

### 정의 내용

|  |
| --- |
| typedef enum {  LDA = 0x00,  LDB = 0x68,  LDT = 0x74,  LDX = 0x04,  LDCH = 0x50,  STA = 0x0C,  STL = 0x14,  STX = 0x10,  STCH = 0x54,  JSUB = 0x48,  JEQ = 0x30,  JGT = 0x34,  JLT = 0x38,  J = 0x3C,  COMP = 0x28,  TD = 0xE0,  RD = 0xD8,  RSUB = 0x4C,  WD = 0xDC,  CLEAR = 0xB4,  COMPR = 0xA0,  TIXR = 0xB8  } Operator; |

## 모듈이름: ADDRESSING\_MODE ENUM

### 기능

Addressing 모드를 enum으로 매핑함.

### 정의 내용

|  |
| --- |
| typedef enum {  ENUM\_IMMEDIATE\_ADDRESSING,  ENUM\_SIMPLE\_ADDRESSING,  ENUM\_INDIRECT\_ADDRESSING,  ENUM\_ADDRESSING\_ERROR  } ADDRESSING\_MODE; |

## 모듈이름: construct\_debugger()

### 기능

Debugger 구조체의 생성자(메모리를 할당받고 초기화를 진행하는) 함수..

### 사용변수

없음.

## 모듈이름: destroy\_debugger(Debugger\*\* debugger)

### 기능

Debugger 구조체와 그와 연관된 변수들에 대한 소멸자(메모리를 해제하는) 함수.

### 사용변수

|  |  |
| --- | --- |
| Debugger\*\* debugger | 해제할 debugger 를 포인팅함 |

## 모듈이름: construct\_registers()

### 기능

Registers 구조체의 생성자 함수.

### 사용변수

없음.

## 모듈이름: destroy\_registers(Registers\*\* registers)

### 기능

Registers 구조체의 소멸자 함수.

### 사용변수

|  |  |
| --- | --- |
| Registers\*\* registers | 해제할 registers 를 포인팅함 |

## 모듈이름: reset\_registers(Registers\* registers)

### 기능

Registers 구조체를 리셋함..

### 사용변수

|  |  |
| --- | --- |
| Registers\* registers | 레지스터 정보를 저장함. |

## 모듈이름: loader\_linker(Debugger \*debugger, Memories \*memories)

### 기능

loader [obj1] [obj2].. 명령을 실질적으로 수행함. Pass1 을 수행하는 함수와 pass2 를 수행하는 함수를 호출하고 성공적으로 load되었다면 load 된정보를 출력해서 보여주는 함수를 호출한다.

### 사용변수

|  |  |
| --- | --- |
| Debugger \*debugger | 디버깅, load, run 등의 명령 수행을 위한 정보들을 저장함.. |
| Memories \*memories | 메모리 정보를 저장함 |

## 모듈이름: run(Debugger \*debugger, Memories \*memories)

### 기능

run 명령을 실질적으로 수행함. Load 된 프로그램을 실행하고 중간에 브레이크포인트 가 있으면 이를 적절히 핸들링한다.

### 사용변수

|  |  |
| --- | --- |
| Debugger \*debugger | 디버깅, load, run 등의 명령 수행을 위한 정보들을 저장함.. |
| Memories \*memories | 메모리 정보를 저장함 |
| Registers\* registers | 레지스터 정보를 저장함. |
| uint32\_t tmp | Tmp 변수 |
| bool is\_continue | Continue 할지 여부를 저장함 |
| uint8\_t opcode | Opcode 를 저장함 |
| uint32\_t instruction\_val | Instruction의 값을 저장함. |
| uint8\_t memory\_val | 메모리 값을 저장함 |
| int instruction\_size | Instruction 의 크기를 저장함 |

## 모듈이름: loader\_linker\_pass1(Debugger \*debugger)

### 기능

Loader 의 Pass1 과정을 수행한다..

### 사용변수

|  |  |
| --- | --- |
| Debugger \*debugger | 디버깅, load, run 등의 명령 수행을 위한 정보들을 저장함.. |
| int csaddr | CSADDR 정보를 저장한다. |

## 모듈이름: loader\_linker\_pass1\_one(Debugger \*debugger, int file\_num, int \*csaddr)

### 기능

Loader 의 Pass1 과정중에 파일 하나에 대한 작업을 수행한다..

### 사용변수

|  |  |
| --- | --- |
| Debugger \*debugger | 디버깅, load, run 등의 명령 수행을 위한 정보들을 저장함.. |
| int\* csaddr | CSADDR 정보를 저장한다. |
| Int file\_num | 파일 번호를 저장한다. |
| FILE \*fp | 파일 포인터 |
| char buf[1010] | 버퍼 정보를 저장함 |
| bool is\_header | 헤더가 있는지 여부를 저장함 |
| int base\_address | base가 되는 주소를 저장함. 원래의 csaddr 주소 |

## 모듈이름: loader\_linker\_pass2(Debugger \*debugger, Memories \*memories)

### 기능

Loader 의 Pass2 과정을 수행한다...

### 사용변수

|  |  |
| --- | --- |
| Debugger \*debugger | 디버깅, load, run 등의 명령 수행을 위한 정보들을 저장함.. |
| int csaddr | CSADDR 정보를 저장한다. |
| Memories \*memories | 메모리 정보를 저장한다. |

## 모듈이름: loader\_linker\_pass2\_one(Debugger \*debugger, Memories \*memories, int file\_num , int \*csaddr)

### 기능

Loader 의 Pass2 과정중 파일 하나에 대한 작업을 수행한다...

### 사용변수

|  |  |
| --- | --- |
| Debugger \*debugger | 디버깅, load, run 등의 명령 수행을 위한 정보들을 저장함.. |
| int\* csaddr | CSADDR 정보를 저장한다. |
| Int file\_num | 파일 번호를 저장한다. |
| Memories \*memories | 메모리 정보를 저장한다. |
| FILE\* fp | 파일 포인터 |

## 모듈이름: construct\_load\_info\_list()

### 기능

Load\_info\_list 구조체의 생성자 함수

### 사용변수

없음

## 모듈이름: print\_load\_infos(LoadInfoList \*load\_infos)

### 기능

Load 된 정보를 출력해줌

### 사용변수

|  |  |
| --- | --- |
| LoadInfoList\* load\_infos | 로드된 정보를 저장함 |
| Unsigned int total\_length | 총 할당된 길이를 저장 |
| Const int count | 로드된 정보의 개수를 저장함 |

## 모듈이름: execute\_operator(Debugger \*debugger, Memories \*memories, Instruction \*instruction)

### 기능

Instruction에 대응되는 Operator(LDA, LDB, 등등)를 실행함

### 사용변수

|  |  |
| --- | --- |
| Debugger \*debugger | 디버깅, load, run 등의 명령 수행을 위한 정보들을 저장함.. |
| Memories \*memories | 메모리 정보를 저장한다. |
| Instruction \*instruction | Instruction 정보를 저장함. |
| Registers\* registers | 레지스터 정보를 저장한다. |
| uint32\_t value,val1,val2 | 값 정보를 저장하기 위한 변수 |
| static size\_t device\_input\_idx | Device Read Write 구현을 위한 변수 |
| char inputDevice[] | 가상 device 변수 |

## 모듈이름: calculate\_TA(Instruction\* instruction, Registers\* registers)

### 기능

Target Address 를 계산하여 리턴함.

### 사용변수

|  |  |
| --- | --- |
| Instruction \*instruction | Instruction 정보를 저장함. |
| Registers\* registers | 레지스터 정보를 저장한다. |
| uint32\_t TA | Target address 정보를 저장한다. |
| uint32\_t b,p,x | B,P,X 레지스터의 값을 저장함 |
| uint32\_t address | 주소를 저장함. |

## 모듈이름: handling\_bp(Debugger \*debugger, int instruction\_size)

### 기능

브레이크 포인트를 핸들링함. .

### 사용변수

|  |  |
| --- | --- |
| Debugger \*debugger | 디버거 정보를 저장함 |
| Int instruction\_size | Instruction 크기를 저장함 |
| Bool is\_break | 브레이크 포인트가 있는지 여부를 저장함 |
| Unsigned int bp | Bp의 주소를 저장함 |

# 전역 변수 정의

## long income[100]

수입을 저장한다.

## int month[100], day[100], year[100]

생년월일을 저장한다.

## int ctr

int ctr – 사용자의 입력을 받는 loop counter로 입력된 명수를 저장한다.

.

## int cont

Continue \_funtion에서 사용자의 입력을 받는데 사용된다.(1이면 프로그램계속 수행, 0이면 종료)

# 코드

/\*포함되는 파일\*/

#include<stdio.h>

/\*정의되는 상수\*/

#define MAX 100

#define YES 1

#define NO 0

/\*변수\*/

long income[MAX];

int month[MAX], day[MAX], year[MAX];

int cont;

int ctr;

/\*함수 원형\*/

void main(void);

int display\_instructions(void);

void get\_data(void);

void display\_report(void);

int continue\_funtion(void);

/\*프로그램 시작\*/

void main(void){

cont = display\_instructions();

if (cont == YES){

get\_data();

display\_report();

}else

printf("\nProgram Aborted by User!!\n\n");

}

/\*------------------------------------------------------------------------------------\*/

/\*함수 : display\_intructions\*/

/\*목적 : 프로그램 사용에 대한 정보를 나타내고, 중단하고 싶으면 0을 계속하고 싶으면 1을 입력할것을 사용자에게 요청한다.\*/

/\*리턴값 : YES - 1

NO - 0\*/

/\*------------------------------------------------------------------------------------\*/

int display\_instructions(void){

printf("\n\n");

printf("\nThis program enables you to enter up to 99 people\'s");

printf("\nincomes and birthdays. It then p;rints the incomes by");

printf("\nmonth along with the overall income and overall average.");

cont = continue\_funtion();

return(cont);

}

/\*------------------------------------------------------------------------------------\*/

/\*함수 : get\_data()\*/

/\*목적 : 사용자로부터 데이터를 입력받는다.

100명의 사용자가 입력되거나 사용자가 월에대해 0을 입력할때까지 계속해서 데이터를 입력받는다\*/

/\*리턴값 : 없음\*/

/\*------------------------------------------------------------------------------------\*/

void get\_data(void){

for(cont = YES, ctr = 0; ctr < MAX && cont == YES; ctr++){

printf("\nEnter information for Person %d.", ctr + 1 );

printf("\n\tenter Birthday: ");

do{

printf("\n\tMonth(0 - 12) : ");

scanf("%d", &month[ctr]);

}while(month[ctr] < 0 || month[ctr] > 12);

do{

printf("\n\tday(0 - 31) : ");

scanf("%d", &day[ctr]);

}while(day[ctr] < 0 || day[ctr] > 31);

do{

printf("\n\tYear(0 - 2006) : ");

scanf("%d", &year[ctr]);

}while(year[ctr] < 0 || year[ctr] > 2006);

printf("\nEnter Yearly Income (whole dollars): ");

scanf("%d", &income[ctr]);

cont = continue\_funtion();

}

}

/\*------------------------------------------------------------------------------------\*/

/\*함수 : display\_report\*/

/\*목적 : 화면에 보고서를 나타낸다.\*/

/\*리턴값 : 없음\*/

/\*------------------------------------------------------------------------------------\*/

void display\_report(){

int x, y;

long grand\_total = 0, month\_total = 0;

printf("\n\n");

printf("\n SALARY SUMMARY");

printf("\n ==============");

for(x = 0; x <= 12; x++){

month\_total = 0;

for(y = 0; y <ctr; y++){

if(month[y] == x)

month\_total += income[y];

}

printf("\n\nTotal for month %d is %ld",x, month\_total);

grand\_total += month\_total;

}

printf("\n\nReport totals : ");

printf("\n\nTotal income is %d", grand\_total);

printf("\n\n\* \* \* End of Report \* \* \*\n");

}

/\*------------------------------------------------------------------------------------\*/

/\*함수 : continue\_funtion()\*/

/\*목적 : 사용자가 계속 진행하고 싶은지의 여부를 묻는다.\*/

/\*리턴값 : YES - 계속하고 싶은 경우

NO - 중단하고 싶은 경우\*/

/\*------------------------------------------------------------------------------------\*/

int continue\_funtion(void){

int ask;

printf("\n\nDo you wish to continue? (0 = NO/ 1 = YES) : ");

scanf( "%d", &ask);

while(ask < 0 || ask >1){

printf("\n\n%d is invalid! ", cont);

printf("\n\nPlease enter 0 to Quit or 1 to continue: ");

scanf("%d", &ask);

}

if(ask == 0)

return(NO);

else

return(YES);

}