逢甲大學 106 年下學期

106-2 系統程式組譯器實作 SIC/XE

指導老師：黃志銘

D0542914 資訊二甲 劉祐炘

2018/6/11

目錄

一、 所使用的雜湊演算法

雜湊函數：符號名稱 mod 質數表大小 = 鏈結赫序排序(下圖為示意圖)

```
int HashSize = 11;//prime hash table size
int length = countTotalAscii("CLOOP");
// 'C' + 'L' + 'O' + 'O' + 'P' = 378
=> HashTable[378 % 11] = HashTable[4]
HashTable[0]-> WLOOP → RDREC → NULL
HashTable[1]-> ENDFIL → NULL
HashTable[2]-> NULL
HashTable[3]-> NULL
HashTable[4]-> CLOOP → NULL
   ⋮
HashTable[9]-> NULL
HashTable[10]-> NULL
```

二、 程式碼說明：

在 main 裡的 buildSymTab("input.txt")，修改要讀的檔案即可。(其他註解)

```
686 int main(){//一次讀入一筆，其他麻煩請註解
687     //buildSymTab("srcpro2.6.txt"); //Fig.2.6
688     //buildSymTab("srcpro2.9.txt");  //Fig.2.9
689     buildSymTab("srcpro2.11.txt"); //Fig.2.11
690     buildDestination();
691     buildRecord();
692     //------------------
693     printNodes(Head); //Original Program + Literal Pool.
694     printOpTab(); //OPTAB
695     printSymTab(); //SYMTAB
696     printLitTab(); //LITTAB
697     printRegTab(); //REGTAB
698     printRecord(); //Target program Record
699     printBlocks(use); //Use Block
700     return 0;
701 }
```

三、 程式碼列印：

https://goo.gl/1edeoL

```c
/*
2018/06/12
assembler SIC/XE.
Feng Chia University IECS.
D0542914 Liou Yow Shin
*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#define SIC_XE_REGS 9
#define SIC_XE_OPS 20
#define PRIME_TABLES 11  //Prime hash table Size
typedef struct _OpTab{
    char name[10];
    char p_info[6];
    char format[5];
    char code[3];
}OpTab;
typedef struct _Block_Locctr* Use;
typedef struct _Block_Locctr{
    int address;
    int counter;
    char name[10];
    int key;
    Use next;
}block_locctr;
typedef struct _listNode* listPointer;
typedef struct _listNode{
    char symname[11];  //symbom name
    char exformat;  //extent format-4
    char opcode[10]; //Opcode
    char optag; // (#, @, =
    char optr_1[10]; //op-1
    char optr; //+ - * /
    char optr_2[10]; //op-2
    int address;
    Use use_block;
    char str[50];
    char destination[20];
    listPointer next;
}listNode;
OpTab optab[] = {
    {"STL","m","3/4","14"},
    {"LDB","m","3/4","68"},
    {"JSUB","m","3/4","48"},
    {"LDA","m","3/4","00"},
    {"COMP","m","3/4","28"},
    {"JEQ","m","3/4","30"},
    {"J","m","3/4","3C"},
    {"STA","m","3/4","0C"},
```

```
51          {"CLEAR","r1","2","B4"},
52          {"LDT","m","3/4","74"},
53          {"TD","m","3/4","E0"},
54          {"RD","m","3/4","D8"},
55          {"COMPR","r1,r2","2","A0"},
56          {"STCH","m","3/4","54"},
57          {"TIXR","r1","2","B8"},
58          {"JLT","m","3/4","38"},
59          {"STX","m","3/4","10"},
60          {"LDCH","m","3/4","50"},
61          {"WD","m","3/4","DC"},
62          {"RSUB","null","3/4","4C"},
63      };
64      char FileName[20];
65      char regs_name[][4] =
        {"A","X","L","PC","SW","B","S","T","F"};
66      int regs_number[] = {0,1,2,8,9,3,4,5,6};
67      listPointer SymTab[PRIME_TABLES];
68      listPointer LitTab[PRIME_TABLES];
69      listPointer Head;
70      listPointer pool;
71      listPointer Record;
72      Use use;
73      int base,pc;
74      int use_cnt = 1;
75      int searchOpTab(char* opname){
76          int i;
77          for(i=0;i<SIC_XE_OPS;i++){
78              if(!strcmp(optab[i].name,opname))return i;
79          }
80          return -1;
81      }
82      char* getStr(char* str,int start,int end){
83          char* temp;
84          temp = (char*)malloc(sizeof(char)*(end-start+1));
85          int index=0,i;
86          for(i=start;i<=end;i++){
87              if(str[i]==' '|| i >= strlen(str) || str[i] ==
                '\n')break;
88              temp[index++] = str[i];
89          }
90          temp[index] = '\0';
91          return temp;
92      }
93      listPointer createNode(){
94          listPointer newptr = NULL;
95          newptr = (listPointer)malloc(sizeof(listNode));
96          newptr->next = NULL;
97          if(newptr)return newptr;
98          else exit(-1);
```

```
 99  }
100  Use createBlock(){
101      Use newptr = NULL;
102      newptr = (Use)malloc(sizeof(block_locctr));
103      newptr->next = NULL;
104      newptr->counter = 0;
105      if(newptr)return newptr;
106      else exit(-1);
107  }
108  void addpool(char* str,Use u){
109      listPointer newptr = createNode();
110      newptr->use_block = u;
111      newptr->exformat = '=';
112      memset(newptr->destination,'
         ',sizeof(newptr->destination));
113      newptr->destination[strlen(newptr->destination)-2] =
         '\0';
114      memset(newptr->str,' ',sizeof(newptr->str));
115      int i ;
116      char *s = newptr->str;
117      s[0] = '*';
118      for(i=0;i<8;i++){
119          s[i+7] = str[i+15];
120      }
121      strcpy(newptr->symname,getStr(str,16,23));
122      if(pool == NULL){
123          pool = newptr;
124      }else{
125          listPointer t = pool;
126          while(t->next != NULL)t = t->next;
127          t->next = newptr;
128      }
129  }
130  void addLiteral(listPointer *head, char* str,Use u){
131      listPointer newptr = createNode();
132      newptr->use_block = u;
133      memset(newptr->str,' ',strlen(str));
134      int i ;
135      char *s = newptr->str;
136      s[0] = '*';
137      for(i=0;i<8;i++){
138          s[i+7] = str[i+15];
139      }
140      //s[i] = '\0';
141      strcpy(newptr->symname,getStr(str,16,23));
142      if(*head == NULL){
143          *head = newptr;
144      }else{
145          listPointer t = *head;
146          while(t->next!=NULL)t = t->next;
```

```
147             t->next = newptr;
148         }
149     }
150     Use addBlock(Use *head, char *str){
151         Use newptr = createBlock();
152         newptr->key = use_cnt++;
153         strcpy(newptr->name,getStr(str,16,23));
154         if(*head == NULL){
155             *head = newptr;
156         }else{
157             Use t = *head;
158             while(t->next!=NULL)t = t->next;
159             t->next = newptr;
160         }
161         return newptr;
162     }
163     listPointer addNode(listPointer *head, char* str,Use u){
164         listPointer newptr = createNode();
165         newptr->exformat = str[7];
166         strcpy(newptr->opcode,getStr(str,8,13));
167         newptr->optag = str[15];
168         newptr->optr = str[24];
169         strcpy(newptr->optr_1,getStr(str,16,23));
170         strcpy(newptr->optr_2,getStr(str,25,32));
171         strcpy(newptr->str,str);
172         strcpy(newptr->symname,getStr(str,0,5));
173         newptr->use_block = u;
174         newptr->address = u->counter;
175         newptr->destination[0] = '\0';
176         if(*head == NULL){
177             *head = newptr;
178         }else{
179             listPointer t = *head;
180             while(t->next!=NULL)t = t->next;
181             t->next = newptr;
182         }
183         return newptr;
184     }
185     void addSymTab(listPointer* head, listPointer ptr){
186         listPointer newptr = createNode();
187         newptr->address = ptr->address;
188         newptr->use_block = ptr->use_block;
189         strcpy(newptr->symname,ptr->symname);
190         if (*head == NULL) {
191             *head = newptr;
192         }
193         else {
194             listPointer temp = *head;
195             while(temp->next != NULL) {
196                 temp = temp->next;
```

```
197             }
198             temp->next = newptr;
199         }
200     }
201     void addRecord(listPointer *h,char *s){
202         listPointer newptr = createNode();
203         strcpy(newptr->str,s);
204         if(*h == NULL)*h = newptr;
205         else{
206             listPointer t = *h;
207             while(t->next != NULL){
208                 t = t->next;
209             }
210             t->next = newptr;
211         }
212     }
213     int searchRegs(char* Regname){
214         int i;
215         for(i=0;i<SIC_XE_REGS;i++){
216             if(!strcmp(regs_name[i],Regname))return
              regs_number[i];
217         }
218         return 0;
219     }
220     listPointer searchSymTab(listPointer t,char* searchName){
221         while(t!=NULL) {
222             if(!strcmp(t->symname,searchName)){
223                 return t;
224             }
225             else t = t->next;
226         }
227         return NULL;
228     }
229     Use searchBlock(Use t,char* name){
230         while(t!=NULL){
231             if(!strcmp(t->name,name))return t;
232             else t = t->next;
233         }
234         return NULL;
235     }
236     listPointer searchLitTab(listPointer t, char* searchName){
237         while(t!=NULL){
238             if(!strcmp(t->symname,searchName)){
239                 return t;
240             }else{
241                 t = t->next;
242             }
243         }
244         return NULL;
245     }
```

```
246  void clearpool(Use u){
247      int symIndex;
248      listPointer n;
249      while(pool!=NULL){
250          symIndex = countTotalAscii(pool->symname) %
         PRIME_TABLES;
251          n = searchLitTab(LitTab[symIndex],pool->symname);
252          pool->address = u->counter;
253          n->address = u->counter;
254          n->use_block = u;
255          pool->use_block = u;
256          if(pool->symname[0]=='C'){
257              u->counter += strlen(pool->symname) - 3;
258          }else if(pool->symname[0]=='X'){
259              u->counter += (strlen(pool->symname) - 3) / 2;
260          }
261          //listPointer d = pool;
262          pool = pool->next;
263          //free(d);
264      }
265      pool = NULL;
266  }
267
268  int countTotalAscii(char* str){
269      int i,total=0;
270      for(i=0;i<strlen(str);i++){
271          total += str[i];
272      }
273      return total;
274  }
275  char* hexToStr6(int temp){
276      int index=5;
277      char* ar;
278      ar = (char*)malloc(sizeof(char)*7);
279      memset(ar,'0',6);
280      while(temp>0){
281          if (temp % 16 >= 10){
282              ar[index] = temp % 16 + 55;
283          }else{
284              ar[index] = temp % 16 + 48;
285          }
286          temp /= 16;
287          index--;
288      }
289      return ar;
290  }
291  void toUpper(char *s){
292      int i=0;
293      while(s[i]){
294          if(s[i] >= 97 && s[i] <= 122){
```

```
295                 s[i] -= 32;
296             }
297             i++;
298         }
299     }
300     void buildSymTab(char* filename){
301         sprintf(FileName,"%s",filename);
302         FILE  *f = fopen(filename, "r");
303         if(f==NULL){
304             printf("no file.\n");
305             exit(-1);
306         }
307         char tempc;
308         char str[50];
309         int index=0,symIndex;
310         int cnt=0,result;
311         int op_len,r;  //指令長度
312         Use u;
313         listPointer n,opt1,opt2;
314         use = createBlock();
315         strcpy(use->name,"DEFAULT");
316         use->key = 0;
317         u = use;
318         memset(str,' ',strlen(str));
319         while(1){
320             result = fscanf(f, "%c", &tempc);
321             if(tempc!='\n' && tempc!='\0' && result!=EOF){
322                 str[index] = tempc;
323                 index++;
324             }
325             else{
326                 str[index] = '\0';
327                 index = 0;
328                 //printf("%s\n",str);
329                 if(str[0] == '.')continue;   //註解
330                 if(!strcmp(getStr(str,8,13),"USE")){  //find
                    use ,switch block.
331                     if(strlen(getStr(str,16,23))==0){
332                         u = use;
333                     }else{
334                         u = searchBlock(use,getStr(str,16,23));
335                         if(u == NULL){
336                             u = addBlock(&use,str);
337                         }
338                     }
339                 }
340                 listPointer newptr = addNode(&Head,str,u);
341                 if(newptr->optag == '='){ //       find literal
342                     symIndex =
                        countTotalAscii(newptr->optr_1) %
```

```
                PRIME_TABLES;
343             n =
                searchLitTab(LitTab[symIndex],newptr->optr_
                1);
344             if(n == NULL){
345                 addLiteral(&LitTab[symIndex],str,u);
346                 addpool(str,u);
347                 //printf("have=%s\n",str);
348             }
349         }
350         if(!strcmp(getStr(str,8,13),"LTORG")){
351             newptr->next = pool;
352             clearpool(u);
353         }
354         if(!strcmp(getStr(str,8,13),"END")){
355             newptr->next = pool;
356             clearpool(u);
357         }
358         if(!strcmp(getStr(str,8,13),"EQU")){
359             symIndex =
                countTotalAscii(newptr->optr_1) %
                PRIME_TABLES;
360             opt1 =
                searchSymTab(SymTab[symIndex],newptr->optr_
                1);
361             symIndex =
                countTotalAscii(newptr->optr_2) %
                PRIME_TABLES;
362             opt2 =
                searchSymTab(SymTab[symIndex],newptr->optr_
                2);
363             if(newptr->optr == '+'){
364                 newptr->address = opt1->address +
                    opt2->address;
365             }else if(newptr->optr == '-'){
366                 newptr->address = opt1->address -
                    opt2->address;
367             }else if(newptr->optr == '*'){
368                 newptr->address = opt1->address *
                    opt2->address;
369             }else if(newptr->optr == '/'){
370                 newptr->address = opt1->address /
                    opt2->address;
371             }
372         }
373         if(!strcmp(getStr(str,8,13),"START")){
374             //reset locctr
375             newptr->address = atoi(newptr->optr_1);
376             u->counter = atoi(newptr->optr_1);
377             u->address = atoi(newptr->optr_1);
```

```
                    //start address
378             }else{
379                 r = searchOpTab(getStr(str,8,13));
380                 if(r!=-1){  //find opcode
381                     if(optab[r].format[0]=='2')op_len = 2;
382                     else if(str[7] == '+')op_len = 4;
383                     else op_len = 3;
384                 }else if(!strcmp(getStr(str,8,13),"WORD")){
385                     op_len = 3;
386                 }else if(!strcmp(getStr(str,8,13),"RESW")){
387                     op_len = 3 * atoi(getStr(str,16,23));
388                 }else if(!strcmp(getStr(str,8,13),"RESB")){
389                     op_len = atoi(getStr(str,16,23));
390                 }else
                    if(!strcmp(getStr(str,8,13),"BYTE")){
391                     op_len = strlen(getStr(str,16,23)) - 3;
392                     if(str[16]=='X')op_len /= 2;
393                 }
394                 if(str[0]!=' ') { //have Symbom
395                     symIndex =
                        countTotalAscii(newptr->symname) %
                        PRIME_TABLES;
396                     n =
                        searchSymTab(SymTab[symIndex],newptr->s
                        ymname);
397                     if(n == NULL){
398
                            addSymTab(&SymTab[symIndex],newptr)
                            ;
399                         //printf("%s\n",newptr->symname);
400                     }else{
401                         printf("error:repeat symname!\n");
402                         exit(-1);
403                     }
404                 }
405                 u->counter += op_len;
406                 op_len = 0;
407             }
408             if(result == EOF)break;
409         }
410     }
411     int count=use->address;
412     u = use;
413     while(u!=NULL){
414         u->address = count;
415         count += u->counter;
416         u = u->next;
417     }
418 }
419 void buildDestination(){
```

```
420         listPointer p = Head;
421         listPointer n,literal;
422         int symIndex,opIndex;
423         int disp,c,i,xbpe;
424         char temp[8],dispStr[8];
425         while(p!=NULL){
426             if(!strcmp(p->opcode,"BASE")){
427                 symIndex = countTotalAscii(p->optr_1) %
                    PRIME_TABLES;
428                 n = searchSymTab(SymTab[symIndex],p->optr_1);
429                 base = n->address;
430             }else{
431                 opIndex = searchOpTab(p->opcode);
432                 if(opIndex != -1) {//find opcode
433                     c =
                        (int)strtol(optab[opIndex].code,NULL,16);
434                     if(p->optag == '#' ){
435                             c += 1;
436                     }else if(p->optag == '@'){
437                             c += 2;
438                     }else{
439                             c += 3;
440                     }
441                     // set program conter
442                     if(p->exformat == '+'){
443                         pc = p->address + 4;
444                     }else if (optab[opIndex].format[0] == '3'){
445                         pc = p->address + 3;
446                     }else{
447                         pc = p->address + 2;
448                     }
449                     if(p->optag != '='){ //search sym if not
                        a literal
450                         symIndex = countTotalAscii(p->optr_1)
                            % PRIME_TABLES;
451                         n =
                            searchSymTab(SymTab[symIndex],p->optr_1
                            );
452                     }else{
453                         symIndex = countTotalAscii(p->optr_1)
                            % PRIME_TABLES;
454                         n =
                            searchLitTab(LitTab[symIndex],p->optr_1
                            );
455                     }
456                     // start calc
457                     xbpe = 0;
458                     if(p->exformat == '+'){ //format = 4
459
                            if(n)sprintf(temp,"%.2x1%05x",c,n->addr
```

```
                    ess);
460                 else
                    sprintf(temp,"%.2x1%05x",c,atoi(p->optr
                    _1));
461                 strcpy(p->destination,temp);
462             }else{
463                 if(optab[opIndex].format[0] == '3'){
                    //format = 3
464                     if(n){
465                         disp = n->address +
                        n->use_block->address - pc;
466                         if(disp > 2047 || disp < -
                        2048){
467                             xbpe += 4; // TA - base
468                             disp = n->address - base;
469                         }else{
470                             xbpe += 2; //TA - pc
471                         }
472                     }
473                     sprintf(dispStr,"%.3x",disp);
474                     if(disp<0){
475
                        sprintf(dispStr,"%s",dispStr+5)
                        ;
476                     }
477                     if(!strcmp(p->optr_2,"X")){
478                         xbpe += 8;
479                     }
480                     if(n == NULL){//is data ,e.g.#3
481                         i = atoi(p->optr_1);
482                         sprintf(dispStr,"%.3x",i);
483                     }
484
                    sprintf(temp,"%.2x%x%s",c,xbpe,disp
                    Str);
485                     strcpy(p->destination,temp);
486                 }else{ // format 2
487                     c-=3;
488
                    sprintf(temp,"%.2x%x%x",c,searchReg
                    s(p->optr_1)
489                     ,searchRegs(p->optr_2));
490                     strcpy(p->destination,temp);
491                 }
492             }
493         }else if(p->exformat == '='){
494             c = 0;
495             for(i=0;i<strlen(p->symname)-3;i++){
496                 if(p->symname[0] == 'C'){
497                     p->destination[c++] =
```

```
                                p->symname[i+2] / 16 + '0';
498                             p->destination[c++] =
                                (p->symname[i+2] % 16) >=10 ?
499                              p->symname[i+2] % 16 + 55 :
                                 p->symname[i+2] % 16 + 48;
500                         }else{
501                             p->destination[c++] =
                                p->symname[i+2];
502                         }
503                     }
504                     p->destination[c] = '\0';
505                 }else if(!strcmp(p->opcode,"BYTE")){
506                     c = 0;
507                     for(i=0;i<strlen(p->optr_1)-3;i++){
508                         if(p->optr_1[0] == 'C'){
509                             p->destination[c++] =
                                p->optr_1[i+2] / 16 + '0';
510                             p->destination[c++] =
                                (p->optr_1[i+2] % 16) >=10 ?
511                              p->optr_1[i+2] % 16 + 55 :
                                 p->optr_1[i+2] % 16 + 48;
512                         }else{
513                             p->destination[c++] =
                                p->optr_1[i+2];
514                         }
515                     }
516                     p->destination[c] = '\0';
517                 }else if(!strcmp(p->opcode,"WORD")){
518
                        sprintf(p->destination,"%.6x",atoi(p->optr_
                        1));
519                 }
520             }
521             if(strlen(p->destination) != 0){
522                 //printf("%s\n",p->destination);
523                 toUpper(p->destination);
524             }
525             p = p->next;
526         }
527     }
528     void buildRecord(){
529         int codeSize = 0;
530         Use u = use;
531         while(u!=NULL){
532             codeSize += u->counter;
533             u = u->next;
534         }
535         if(Head == NULL)return;
536         listPointer hptr = createNode();
537
```

```
        sprintf(hptr->str,"H%-6s%-6s%-6s",Head->symname,hexToSt
        r6(Head->address)
538     ,hexToStr6(codeSize));
539     //printf("%s\n",hptr->str);   //generate H record.
540     Record = hptr;
541     // Generate T record
542     listPointer ptr = Head;
543     listPointer first = NULL;
544     listPointer Mrecord = NULL;
545     listPointer n;
546     int symIndex;
547     char str[100];
548     strcpy(str,"");
549     char temp[10];
550     char record[100];
551     int count=0;
552     int nextline = 1;
553     while(ptr!=NULL){
554         if(strlen(ptr->destination) != 0){
555             if(nextline == 1){
556                 first = ptr;
557                 nextline = 0;
558             }
559             if(ptr->exformat == '+' && use->next == NULL
                )){// m record
560                 symIndex = countTotalAscii(ptr->optr_1) %
                    PRIME_TABLES;
561                 n =
                    searchSymTab(SymTab[symIndex],ptr->optr_1);
562                 if(n){
563                     sprintf(temp,"M%.6x05",ptr->address +
                        1);
564                     addRecord(&Mrecord,temp);
565                 }
566                 //printf("%s\n",temp);
567             }
568             strcat(str,ptr->destination);
569             count += strlen(ptr->destination) / 2;
570             if(use->next != NULL){
571                 if(ptr->use_block->key !=
                    ptr->next->use_block->key
572                 || strcmp(ptr->next->opcode,"END")
                    ==0)nextline = 1;
573             }
574             if(ptr->next == NULL){
575                 nextline = 1;
576             }
577             if(count >= 29 || nextline == 1){
578
                    sprintf(temp,"T%.6x%.2x",first->address+fir
```

```
                    st->use_block->address,count);
579                 toUpper(temp);
580                 sprintf(record,"%s%s",temp,str);
581                 //printf("%s\n\n",record);
582                 addRecord(&Record,record);
583                 //reset
584                 strcpy(str,"");
585                 count = 0;
586                 nextline = 1;
587             }
588         }
589         ptr = ptr->next;
590     }
591     if(strlen(str) != 0){
592
        sprintf(temp,"T%.6x%.2x",first->address+first->use_
        block->address,count);
593         toUpper(temp);
594         sprintf(record,"%s%s",temp,str);
595         addRecord(&Record,record);
596     }
597     //Mrecord
598     ptr = Record;
599     while(ptr->next != NULL)ptr = ptr->next;
600     ptr->next = Mrecord;
601     //Erecord
602     int index;
603     ptr = Head;
604     while(ptr!=NULL){
605         index = searchOpTab(ptr->opcode);
606         if(index != -1){ //find first op
607             sprintf(temp,"E%.6x",ptr->address);
608             break;
609         }
610         ptr = ptr->next;
611     }
612     addRecord(&Record,temp);
613 }
614 void printRecord(){
615     printf("\nFilename: %s\n",FileName);
616     printf("------------【Record】---------------\n");
617     listPointer t = Record;
618     while(t != NULL){
619         printf("%s\n",t->str);
620         t = t->next;
621     }
622 }
623 void printOpTab(){
624     //printf("\nFilename: %s\n",FileName);
625     printf("\n------------【OPTAB】---------------\n");
```

```
626         int i;
627         printf("Row\tOp_Name\tFortmat\tOpCode\tinfo\n");
628         for(i=0;i<SIC_XE_OPS;i++){
629             printf("%2d\t%s\t%s\t%s\t%s\n",i+1,optab[i].name,
630             optab[i].format,optab[i].code,optab[i].p_info);
631         }
632     }
633     void printRegTab(){
634         //printf("\nFilename: %s\n",FileName);
635         printf("\n------------【REGTAB】---------------\n");
636         int i;
637         printf("Row\tREG_Name\tREG_Code\n");
638         for(i=0;i<SIC_XE_REGS;i++){
639 
                printf("%2d\t%5s\t\t%4d\n",i+1,regs_name[i],regs_nu
                mber[i]);
640         }
641     }
642     void printBlocks(Use t){
643         printf("\nFilename: %s\n",FileName);
644         printf("------------【BLOCK】---------------\n");
645         printf("Name\tKey\taddress\tsize\n");
646         while(t!=NULL){
647 
                printf("%s\t%d\t%04x\t%04x\n",t->name,t->key,t->add
                ress,t->counter);
648             t = t->next;
649         }
650     }
651     void printNodes(listPointer t){
652         int i=1;
653         printf("Filename: %s\n",FileName);
654         printf("-----------【Original Program <Literal
                pool>】--------------------\n");
655         printf("Row/addr/use\tCode\t\t\t\t Target Address\n");
656 
                printf("-----------------------------------------------
                ------------------\n");
657         while(t!=NULL){
658             printf("%2d  %04x
                %d\t%-24s\t\t%s\n",i++,t->address,t->use_block->key
                ,t->str,t->destination);
659             t = t->next;
660         }
661     }
662     void printSymTab(){
663         printf("\nFilename: %s\n",FileName);
664         printf("------------【SYMTAB】---------------\n");
665         int i,row=1;
666         printf("Row Hash\tSymName\t\tAddress\tUse\n");
```

```
667         for(i=0;i<PRIME_TABLES;i++){
668             listPointer t = SymTab[i];
669             while(t != NULL){
670 
                    printf("%2d%4d\t%14s\t%13.4x\t%2d\n",row++,i,t-
                    >symname,t->address,t->use_block->key);
671                 t = t->next;
672             }
673         }
674     }
675     void printLitTab(){
676         printf("\nFilename: %s\n",FileName);
677         printf("------------【LITABLE】---------------\n");
678         printf("Row Hash  LitName  Address    Block\n");
679         int i,row=1;
680         for(i=0;i<PRIME_TABLES;i++){
681             listPointer t = LitTab[i];
682             while(t != NULL){
683                 printf("%2d %3d%12s    %.4x
                    %s\n",row++,i,t->symname,t->address,t->use_bloc
                    k->name);
684                 t = t->next;
685             }
686         }
687     }
688     int main(){//only can read by 1, and other must to be //
689         //buildSymTab("srcpro2.6.txt"); //Fig.2.6
690         //buildSymTab("srcpro2.9.txt");  //Fig.2.9
691         buildSymTab("srcpro2.11.txt"); //Fig.2.11
692         buildDestination();
693         buildRecord();
694         //------------------
695         printNodes(Head); //Original Program + Literal Pool.
696         printOpTab(); //OPTAB
697         printSymTab(); //SYMTAB
698         printLitTab(); //LITTAB
699         printRegTab(); //REGTAB
700         printRecord(); //Target program Record
701         printBlocks(use); //Use Block
702         return 0;
703     }
704 
```

四、 SIC/XE Program

A. 【Input file：srcpro.2.6.txt】

1. 原始程式列印(包括常數池：Literal pool)

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
Filename: srcpro2.6.txt
----------【Original Program <Literal pool>】----------
Row/addr/use    Code                                    Target Address
-------------------------------------------------------------------
 1  0000 0      COPY    START   0
 2  0000 0      FIRST   STL     RETADR                  17202D
 3  0003 0              LDB     #LENGTH                 69202D
 4  0006 0              BASE    LENGTH
 5  0006 0      CLOOP   +JSUB   RDREC                   4B101036
 6  000a 0              LDA     LENGTH                  032026
 7  000d 0              COMP    #0                      290000
 8  0010 0              JEQ     ENDFIL                  332007
 9  0013 0              +JSUB   WRREC                   4B10105D
10  0017 0              J       CLOOP                   3F2FEC
11  001a 0      ENDFIL  LDA     EOF                     032010
12  001d 0              STA     BUFFER                  0F2016
13  0020 0              LDA     #3                      010003
14  0023 0              STA     LENGTH                  0F200D
15  0026 0              +JSUB   WRREC                   4B10105D
16  002a 0              J       @RETADR                 3E2003
17  002d 0      EOF     BYTE    C'EOF'                  454F46
18  0030 0      RETADR  RESW    1
19  0033 0      LENGTH  RESW    1
20  0036 0      BUFFER  RESB    4096
21  1036 0      RDREC   CLEAR   X                       B410
22  1038 0              CLEAR   A                       B400
23  103a 0              CLEAR   S                       B440
24  103c 0              +LDT    #4096                   75101000
25  1040 0      RLOOP   TD      INPUT                   E32019
26  1043 0              JEQ     RLOOP                   332FFA
27  1046 0              RD      INPUT                   DB2013
28  1049 0              COMPR   A       ,S              A004
29  104b 0              JEQ     EXIT                    332008
30  104e 0              STCH    BUFFER  ,X              57C003
31  1051 0              TIXR    T                       B850
32  1053 0              JLT     RLOOP                   3B2FEA
33  1056 0      EXIT    STX     LENGTH                  134000
34  1059 0              RSUB                            4F0000
35  105c 0      INPUT   BYTE    X'F1'                   F1
36  105d 0      WRREC   CLEAR   X                       B410
37  105f 0              LDT     LENGTH                  774000
38  1062 0      WLOOP   TD      OUTPUT                  E32011
39  1065 0              JEQ     WLOOP                   332FFA
40  1068 0              LDCH    BUFFER  ,X              53C003
41  106b 0              WD      OUTPUT                  DF2008
42  106e 0              TIXR    T                       B850
43  1070 0              JLT     WLOOP                   3B2FEF
44  1073 0              RSUB                            4F0000
45  1076 0      OUTPUT  BYTE    X'05'                   05
46  1077 0              END     FIRST
```

2. **操作碼表(OPTAB)列印：**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
------------【OPTAB】---------------
Row     Op_Name Fortmat OpCode  info
 1      STL     3/4     14      m
 2      LDB     3/4     68      m
 3      JSUB    3/4     48      m
 4      LDA     3/4     00      m
 5      COMP    3/4     28      m
 6      JEQ     3/4     30      m
 7      J       3/4     3C      m
 8      STA     3/4     0C      m
 9      CLEAR   2       B4      r1
10      LDT     3/4     74      m
11      TD      3/4     E0      m
12      RD      3/4     D8      m
13      COMPR   2       A0      r1,r2
14      STCH    3/4     54      m
15      TIXR    2       B8      r1
16      JLT     3/4     38      m
17      STX     3/4     10      m
18      LDCH    3/4     50      m
19      WD      3/4     DC      m
20      RSUB    3/4     4C      null
```

3. **符號表(SYMTAB)列印(使用赫序法排列)：**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
Filename: srcpro2.6.txt
-------------【SYMTAB】----------------
Row Hash        SymName         Address Use
 1   0           RLOOP           1040    0
 2   2          BUFFER           0036    0
 3   2           WRREC           105d    0
 4   2          OUTPUT           1076    0
 5   4           INPUT           105c    0
 6   5          ENDFIL           001a    0
 7   5           RDREC           1036    0
 8   5           WLOOP           1062    0
 9   6            EXIT           1056    0
10   7           FIRST           0000    0
11   7           CLOOP           0006    0
12   9             EOF           002d    0
13  10          RETADR           0030    0
14  10          LENGTH           0033    0
```

4. **常數表(LITTAB)列印(使用赫序法排列)：(無常數)**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
Filename: srcpro2.6.txt
------------【LITABLE】---------------
Row Hash  LitName  Address    Block
```

5. **暫存器表(REGTAB)列印：**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
------------【REGTAB】---------------
Row      REG_Name         REG_Code
 1           A               0
 2           X               1
 3           L               2
 4          PC               8
 5          SW               9
 6           B               3
 7           S               4
 8           T               5
 9           F               6
```

6. **目的程式檔(Record)列印：**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
Filename: srcpro2.6.txt
------------【Record】---------------
HCOPY  000000001077
T0000001D17202D69202D4B10103603202629000033200074B10105D3F2FEC032010
T00001D1D0F20160100030F200D4B10105D3E2003454F46B410B400B44075101000
T0010401DE32019332FFADB2013A00433200857C003B8503B2FEA1340004F0000F1
T00105D1AB410774000E32011332FFA53C003DF2008B8503B2FEF4F000005
M00000705
M00001405
M00002705
E000000

Filename: srcpro2.6.txt
------------【BLOCK】---------------
Name    Key     address size
DEFAULT 0       0000    1077

--------------------------------
Process exited after 0.1143 seconds with return value 0
請按任意鍵繼續 . . .
```

B. 【Input file：srcpro.2.9.txt】

1. 原始程式列印(包括常數池：Literal pool)

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
Filename: srcpro2.9.txt
----------【Original Program <Literal pool>】----------
Row/addr/use    Code                                    Target Address
------------------------------------------------------------------
 1   0000 0     COPY    START   0
 2   0000 0     FIRST   STL     RETADR                  17202D
 3   0003 0             LDB     #LENGTH                 69202D
 4   0006 0             BASE    LENGTH
 5   0006 0     CLOOP   +JSUB   RDREC                  4B101036
 6   000a 0             LDA     LENGTH                  032026
 7   000d 0             COMP    #0                      290000
 8   0010 0             JEQ     ENDFIL                  332007
 9   0013 0             +JSUB   WRREC                   4B10105D
10   0017 0             J       CLOOP                   3F2FEC
11   001a 0     ENDFIL  LDA     =C'EOF'                 032010
12   001d 0             STA     BUFFER                  0F2016
13   0020 0             LDA     #3                      010003
14   0023 0             STA     LENGTH                  0F200D
15   0026 0             +JSUB   WRREC                   4B10105D
16   002a 0             J       @RETADR                 3E2003
17   002d 0             LTORG
18   002d 0     *       =C'EOF'                         454F46
19   0030 0     RETADR  RESW    1
20   0033 0     LENGTH  RESW    1
21   0036 0     BUFFER  RESB    4096
22   1036 0     BUFEND  EQU     *
23   1000 0     MAXLEN  EQU     BUFEND  -BUFFER
24   1036 0     RDREC   CLEAR   X                       B410
25   1038 0             CLEAR   A                       B400
26   103a 0             CLEAR   S                       B440
27   103c 0             +LDT    #MAXLEN                 75101000
28   1040 0     RLOOP   TD      INPUT                   E32019
29   1043 0             JEQ     RLOOP                   332FFA
30   1046 0             RD      INPUT                   DB2013
31   1049 0             COMPR   A       ,S              A004
32   104b 0             JEQ     EXIT                    332008
33   104e 0             STCH    BUFFER  ,X              57C003
34   1051 0             TIXR    T                       B850
35   1053 0             JLT     RLOOP                   3B2FEA
36   1056 0     EXIT    STX     LENGTH                  134000
37   1059 0             RSUB                            4F0000
38   105c 0     INPUT   BYTE    X'F1'                   F1
39   105d 0     WRREC   CLEAR   X                       B410
40   105f 0             LDT     LENGTH                  774000
41   1062 0     WLOOP   TD      =X'05'                  E32011
42   1065 0             JEQ     WLOOP                   332FFA
43   1068 0             LDCH    BUFFER  ,X              53C003
44   106b 0             WD      =X'05'                  DF2008
45   106e 0             TIXR    T                       B850
46   1070 0             JLT     WLOOP                   3B2FEF
47   1073 0             RSUB                            4F0000
48   1076 0             END     FIRST
49   1076 0     *       =X'05'                          05
```

2. **操作碼表(OPTAB)列印：**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
------------【OPTAB】---------------
Row     Op_Name Fortmat OpCode  info
 1      STL     3/4     14      m
 2      LDB     3/4     68      m
 3      JSUB    3/4     48      m
 4      LDA     3/4     00      m
 5      COMP    3/4     28      m
 6      JEQ     3/4     30      m
 7      J       3/4     3C      m
 8      STA     3/4     0C      m
 9      CLEAR   2       B4      r1
10      LDT     3/4     74      m
11      TD      3/4     E0      m
12      RD      3/4     D8      m
13      COMPR   2       A0      r1,r2
14      STCH    3/4     54      m
15      TIXR    2       B8      r1
16      JLT     3/4     38      m
17      STX     3/4     10      m
18      LDCH    3/4     50      m
19      WD      3/4     DC      m
20      RSUB    3/4     4C      null
```

3. **符號表(SYMTAB)列印(使用赫序法排列)：**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
Filename: srcpro2.9.txt
-------------【SYMTAB】----------------
Row Hash        SymName         Address Use
 1   0           RLOOP           1040    0
 2   2           BUFFER          0036    0
 3   2           MAXLEN          1000    0
 4   2           WRREC           105d    0
 5   4           INPUT           105c    0
 6   5           ENDFIL          001a    0
 7   5           RDREC           1036    0
 8   5           WLOOP           1062    0
 9   6           EXIT            1056    0
10   7           FIRST           0000    0
11   7           CLOOP           0006    0
12   7           BUFEND          1036    0
13  10           RETADR          0030    0
14  10           LENGTH          0033    0
```

4. **常數表(LITTAB)列印(使用赫序法排列)：**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
Filename: srcpro2.9.txt
------------【LITABLE】---------------
Row Hash  LitName  Address    Block
 1   0      C'EOF'    002d    DEFAULT
 2   3       X'05'    1076    DEFAULT
```

5. **暫存器表(REGTAB)列印：**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
------------【REGTAB】---------------
Row      REG_Name         REG_Code
 1           A               0
 2           X               1
 3           L               2
 4          PC               8
 5          SW               9
 6           B               3
 7           S               4
 8           T               5
 9           F               6
```

6. **目的程式檔(Record)列印：**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
Filename: srcpro2.9.txt
------------【Record】---------------
HCOPY  000000001077
T0000001D17202D69202D4B10103603202629000033200074B10105D3F2FEC032010
T00001D1D0F20160100030F200D4B10105D3E2003454F46B410B400B44075101000
T0010401DE32019332FFADB2013A00433200857C003B8503B2FEA1340004F0000F1
T00105D1AB410774000E32011332FFA53C003DF2008B8503B2FEF4F000005
M00000705
M00001405
M00002705
M00103d05
E000000

Filename: srcpro2.9.txt
------------【BLOCK】---------------
Name    Key     address size
DEFAULT 0       0000    1077

--------------------------------
Process exited after 0.1284 seconds with return value 0
請按任意鍵繼續 . . .
```

C. 【Input file：srcpro.2.11.txt】

1. **原始程式列印(包括常數池：Literal pool)**

D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe

```
Filename: srcpro2.11.txt
----------【Original Program <Literal pool>】--------------------
Row/addr/use    Code                                  Target Address
-------------------------------------------------------------------
 1  0000 0      COPY    START   0
 2  0000 0      FIRST   STL     RETADR                172063
 3  0003 0      CLOOP   JSUB    RDREC                 4B2021
 4  0006 0              LDA     LENGTH                032060
 5  0009 0              COMP    #0                    290000
 6  000c 0              JEQ     ENDFIL                332006
 7  000f 0              JSUB    WRREC                 4B203B
 8  0012 0              J       CLOOP                 3F2FEE
 9  0015 0      ENDFIL  LDA     =C'EOF'               032055
10  0018 0              STA     BUFFER                0F2056
11  001b 0              LDA     #3                    010003
12  001e 0              STA     LENGTH                0F2048
13  0021 0              JSUB    WRREC                 4B2029
14  0024 0              J       @RETADR               3E203F
15  0000 1              USE     CDATA
16  0000 1      RETADR  RESW    1
17  0003 1      LENGTH  RESW    1
18  0000 2              USE     CBLKS
19  0000 2      BUFFER  RESB    4096
20  1000 2      BUFEND  EQU     *
21  1000 2      MAXLEN  EQU     BUFEND  -BUFFER
22  0027 0              USE
23  0027 0      RDREC   CLEAR   X                     B410
24  0029 0              CLEAR   A                     B400
25  002b 0              CLEAR   S                     B440
26  002d 0             +LDT     #MAXLEN               75101000
27  0031 0      RLOOP   TD      INPUT                 E32038
28  0034 0              JEQ     RLOOP                 332FFA
29  0037 0              RD      INPUT                 DB2032
30  003a 0              COMPR   A       ,S            A004
31  003c 0              JEQ     EXIT                  332008
32  003f 0              STCH    BUFFER  ,X            57A02F
33  0042 0              TIXR    T                     B850
34  0044 0              JLT     RLOOP                 3B2FEA
35  0047 0      EXIT    STX     LENGTH                13201F
36  004a 0              RSUB                          4F0000
37  0006 1              USE     CDATA
38  0006 1      INPUT   BYTE    X'F1'                 F1
39  004d 0              USE
40  004d 0      WRREC   CLEAR   X                     B410
41  004f 0              LDT     LENGTH                772017
42  0052 0      WLOOP   TD      =X'05'                E3201B
43  0055 0              JEQ     WLOOP                 332FFA
44  0058 0              LDCH    BUFFER  ,X            53A016
45  005b 0              WD      =X'05'                DF2012
46  005e 0              TIXR    T                     B850
47  0060 0              JLT     WLOOP                 3B2FEF
48  0063 0              RSUB                          4F0000
49  0007 1              USE     CDATA
50  0007 1              LTORG
51  0007 1      *       =C'EOF'                       454F46
52  000a 1      *       =X'05'                        05
53  000b 1              END     FIRST
```

2. **操作碼表(OPTAB)列印：**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
------------【OPTAB】---------------
Row     Op_Name Fortmat OpCode  info
 1      STL     3/4     14      m
 2      LDB     3/4     68      m
 3      JSUB    3/4     48      m
 4      LDA     3/4     00      m
 5      COMP    3/4     28      m
 6      JEQ     3/4     30      m
 7      J       3/4     3C      m
 8      STA     3/4     0C      m
 9      CLEAR   2       B4      r1
10      LDT     3/4     74      m
11      TD      3/4     E0      m
12      RD      3/4     D8      m
13      COMPR   2       A0      r1,r2
14      STCH    3/4     54      m
15      TIXR    2       B8      r1
16      JLT     3/4     38      m
17      STX     3/4     10      m
18      LDCH    3/4     50      m
19      WD      3/4     DC      m
20      RSUB    3/4     4C      null
```

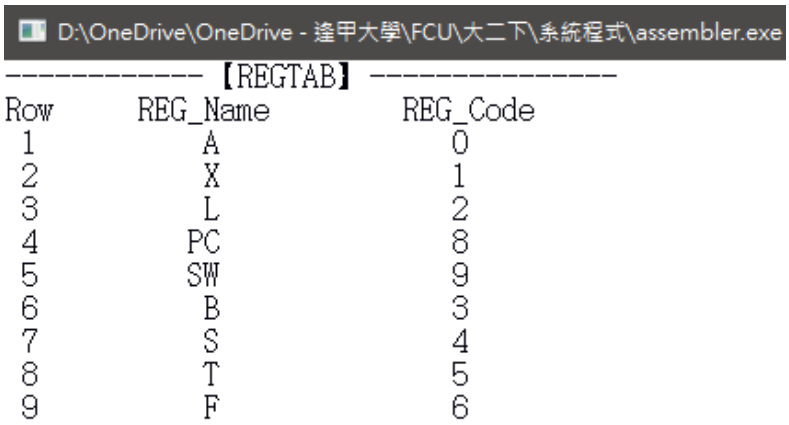
3. **符號表(SYMTAB)列印(使用赫序法排列)：**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
Filename: srcpro2.11.txt
-------------【SYMTAB】-----------------
Row Hash        SymName         Address Use
 1   0           RLOOP           0031    0
 2   2          BUFFER           0000    2
 3   2          MAXLEN           1000    2
 4   2           WRREC           004d    0
 5   4           INPUT           0006    1
 6   5          ENDFIL           0015    0
 7   5           RDREC           0027    0
 8   5           WLOOP           0052    0
 9   6            EXIT           0047    0
10   7           FIRST           0000    0
11   7           CLOOP           0003    0
12   7          BUFEND           1000    2
13  10          RETADR           0000    1
14  10          LENGTH           0003    1
```

4. **常數表(LITTAB)列印(使用赫序法排列)：**

```
D:\OneDrive\OneDrive - 逢甲大學\FCU\大二下\系統程式\assembler.exe
Filename: srcpro2.11.txt
------------【LITABLE】---------------
Row Hash  LitName  Address    Block
 1   0      C'EOF'    0007    CDATA
 2   3       X'05'    000a    CDATA
```

5. **暫存器表(REGTAB)列印：**



6. **目的程式檔(Record)列印：**