系統程式

組譯器實作書面報告

班級：資訊二丙

姓名：蔡聿涵

學號：D0713019

一、程式碼

```
1   /*我的原始敘述是以 TAB 為空白，
2   沒有照著 PPT 的欄位去輸入原始敘述，
3   已經和老師溝通過，在此註明要用附加的.txt 檔才能正確執行。*/
4   #include<stdio.h>
5   #include<stdlib.h>
6   #include<string.h>
7   #define OP 20
8   #define primeTable 11
9   typedef struct OpTable{          //build Table
10          char name[10];
11          char infor[10];
12          char format[5];
13          char code[5];
14          struct OpTable* next;
15  }opTable;
16  typedef struct Locctr* Use;
17  typedef struct Locctr{
18          int size;           //block length
19          int address;        //block start address
20          char name[10]; //block name
21          int num;            //bloc number
22          Use next;
23  }Locctr;
24  typedef struct Node* list;
25  typedef struct Node{
26          char name[10];
27          char extend;     // + ,' '
28          char opcode[10];
29          char mark;       // =,#,@
30          char oper1[10]; //運算元 1
31          char oper;       //分隔號, 運算 + 或 -
32          char oper2[10]; //運算元 2
33          int address;
34          char target[20];
35          char str[80];
36          Use block;
37          list next;
38  };
39  typedef struct Reg{          //build register Table
40          char name[5];
```

```
41          int num;
42      }reg;
43      opTable optab[] = {          //建 opTab
44          {"STL","m","3/4","14"},
45          {"LDB","m","3/4","68"},
46          {"JSUB","m","3/4","48"},
47          {"LDA","m","3/4","00"},
48          {"COMP","m","3/4","28"},
49          {"JEQ","m","3/4","30"},
50          {"J","m","3/4","3C"},
51          {"STA","m","3/4","0C"},
52          {"CLEAR","r1","2","B4"},
53          {"LDT","m","3/4","74"},
54          {"TD","m","3/4","E0"},
55          {"RD","m","3/4","D8"},
56          {"COMPR","r1,r2","2","A0"},
57          {"STCH","m","3/4","54"},
58          {"TIXR","r1","2","B8"},
59          {"LDCH","m","3/4","50"},
60          {"WD","m","3/4","DC"},
61          {"JLT","m","3/4","38"},
62          {"STX","m","3/4","10"},
63          {"RSUB","NULL","3/4","4C"},
64      };
65      reg regtab[] = {
66          {"A",0},
67          {"X",1},
68          {"L",2},
69          {"PC",8},
70          {"SW",9},
71          {"B",3},
72          {"S",4},
73          {"T",5},
74          {"F",6},
75      };
76
77      list symTab[primeTable];
78      list litTab[primeTable];
79      char Fname[30];
80      int use_num = 1;
81      int base, pc;
82      list head = NULL;
```

```
83  list lit_head = NULL;
84  list mHead = NULL;
85  Use useHead = NULL;
86
87  Use buildBlock(char*);
88  void buildLitTab(int, list, Use);
89  void buildSymTab(int, list);
90  void clearList(Use);
91  int Hash(char*);
92  list newnode(void);
93  Use newBlock(void);
94  list setnode(char*);
95  list searchSymTab(list, char*);
96  list searchLitTab(list, char*);
97  Use searchBlock(char*);
98  int searchOpTab(char*);
99  int searchReg(char*);
100 void printPool(void);
101 void printLitTab(void);
102 void printSymTab(void);
103 void printOpTab(void);
104 void printRegTab(void);
105 void onepass(char*);        //輸入原始敘述，建 symTab 、litTab ，算 Address ，算 block
106 void twopass(void);         //算目的碼
107 void Object_program(void);      //算+印 object_program
108
109 void printPool(void){
110         int i = 1;
111         printf("Filename: %s\n", Fname);
112         printf("------------------------------【Literal Pool】------------------------------\n");
113         printf("%3s%8s%6s\t%s\t\t\t\t\t %s\n\n","Row","Address","Block","Code","Target");
114         list ptr = head;
115         while(ptr != NULL){
116                 printf("%-4d %04X%6d\t\t%-6s%5c%-6s%5c%-6s%5c%-s\t %s\n", i, ptr->address, ptr->block->num,
    ptr->name, ptr->extend, ptr->opcode, ptr->mark, ptr->oper1, ptr->oper, ptr->oper2, ptr->target);
117                 ptr = ptr -> next;
118                 i++;
119         }
120 }
121 void printLitTab(void){
122         int i, j = 1;
123         printf("\nFilename: %s\n", Fname);
```

```
124         printf("---------- 【LITTAB】 ----------\n");
125         printf("%4s%10s\t%s\t    %s          \n","Row","LitName","Address","UseName");
126         for(i = 0; i < primeTable; i++){
127             if(litTab[i] != NULL){
128                 list ptr = litTab[i];
129                 while(ptr != NULL){
130                     printf("%4d%10s\t%04X\t    %s\n", j, ptr->opcode, ptr->address, ptr->block->name);
131                     ptr = ptr -> next;
132                     j++;
133                 }
134             }
135         }
136 }
137 void printOpTab(){
138         int i, j = 1;
139         printf("\nFilename: %s\n", Fname);
140         printf("---------- 【OPTAB】 ----------\n");
141         printf("%4s%8s\t%s\t%s\t%s          \n","Row","OpName","Format", "OpCode", "Infor");
142         for(i = 0; i < 20; i++){
143             printf("%4d%8s\t%s\t%s\t%s\n", j, optab[i].name, optab[i].format, optab[i].code, optab[i].infor);
144             j++;
145         }
146 }
147 void printRegTab(){
148         int i, j = 1;
149         printf("\nFilename: %s\n", Fname);
150         printf("---------- 【REGTAB】 ----------\n");
151         printf("%4s %8s\t%s\n","Row","RegName","RegCode");
152         for(i = 0; i < 9; i++){
153             printf("%4d%8s\t%X\n", j, regtab[i].name, regtab[i].num);
154             j++;
155         }
156 }
157 void printSymTab(void){
158         int i, j = 1;
159         printf("\nFilename: %s\n", Fname);
160         printf("---------- 【SYMTAB】 ----------\n");
161         printf("%4s%10s\t%s\t    %s\n","Row","SymName","Address","UseName");
162         for(i = 0; i < primeTable; i++){
163             if(symTab[i] != NULL){
164                 list ptr = symTab[i];
165                 while(ptr != NULL){
```

```
166                             printf("%4d%10s\t%04X\t    %s\n", j, ptr->name, ptr->address, ptr->block->name);
167                             ptr = ptr -> next;
168                             j++;
169                     }
170             }
171     }
172 }
173 Use newBlock(){
174     Use node = (Use)malloc(sizeof(Locctr));
175     node -> next = NULL;
176     node -> size = 0;
177     return node;
178 }
179 list newnode(){
180     list node = (list)malloc(sizeof(struct Node));
181     node -> next = NULL;
182     return node;
183 }
184 list setnode(char* str){            //把 node 做分類並串起來 head
185     list node = newnode();
186     int i = 0, j = 0, flag = 0, temp;
187     char tmp[10];
188     memset(tmp, '\0', 10);
189     if(str[0] == '\t'){       //若開頭為空格 ， 助記碼
190         i = 1;
191     }else{                              //name
192         while(str[i] != '\t' && str[i] != '\0'){
193             tmp[j] = str[i];
194             if(str[i+1] != '\t' && str[i+1] != '\0'){
195                 j++;
196             }
197             i++;
198         }
199         j = 0;
200         i++;
201     }
202     strcpy(node->name, tmp);
203     memset(tmp,'\0',10);
204     while(str[i] != '\t' && str[i] != '\0'){    //助記碼
205         if(str[i] == '+'){
206             node->extend = str[i];
207             flag = 1;
```

```
208             }else{
209                     tmp[j] = str[i];
210                     if(str[i+1] != '\t' && str[i+1] != '\0'){
211                             j++;
212                     }
213             }
214             i++;
215     }
216     if(flag == 0){
217             node -> extend = '\0';
218     }
219     strcpy(node->opcode, tmp);
220     j = 0;
221     flag = 0;
222     //運算碼 oper
223     i++;
224     temp = i;
225     memset(tmp,'\0',10);
226     if(str[temp]=='#' || str[temp]=='@' || str[temp]=='='){
227             node -> mark = str[temp];
228             i++;
229     }else{
230             node -> mark = '\0';
231     }
232     while(str[i] != '\t' && str[i] != '\0'){
233             if(str[i]=='-' || str[i]=='+' || str[i]==','){
234                     strcpy(node->oper1, tmp);
235                     node->oper = str[i];
236                     j = 0;
237                     i++;
238                     memset(tmp,'\0',10);
239                     while(str[i]!='\t' && str[i]!='\0'){
240                             tmp[j] = str[i];
241                             if(str[i+1] != '\t' && str[i+1] != '\0'){
242                                     j++;
243                             }
244                             i++;
245                     }
246                     strcpy(node->oper2, tmp);
247                     flag = 1;
248                     break;
249             }else{
```

```
250                     tmp[j] = str[i];
251                     if(str[i+1] != '\t' && str[i+1] != '\0'){
252                             j++;
253                     }
254                     i++;
255             }
256     }
257     if(flag == 0){
258             strcpy(node->oper1, tmp);
259             memset(tmp, '\0', 10);
260             node -> oper = '\0';
261             strcpy(node->oper2, tmp);
262     }
263     j = 0;
264     flag = 0;
265     memset(tmp,'\0',10);
266     //串起來
267     if(head == NULL){
268             head = node;
269     }else{
270             list ptr = head;
271             while(ptr->next != NULL){
272                     ptr = ptr -> next;
273             }
274             ptr -> next = node;
275     }
276     return node;
277 }
278 int Hash(char* str){
279     int sum = 0 ,i;
280     int len = strlen(str);
281     for(i=0; i<len; i++){
282             sum += str[i];
283     }
284     sum %= primeTable;
285     return sum;
286 }
287 void buildLitTab(int index, list ptr, Use use){
288     int flag = 0;
289     list node = newnode();                  //存到LitTab
290     node -> extend = '=';
291     strcpy(node->opcode, ptr->oper1);
```

```
292         node -> block = use;
293         list temp = newnode();                  //存到Lit_head
294         strcpy(temp->name, "*");
295         temp -> extend = '=';
296         strcpy(temp->opcode, node->opcode);
297         temp -> mark = '\0';
298         strcpy(temp->oper1, "\0");
299         temp -> oper = '\0';
300         strcpy(temp->oper2, "\0");
301         temp -> block = use;
302         if(litTab[index] == NULL){
303             litTab[index] = newnode();
304             litTab[index] = node;
305         }else{
306             list tmp = litTab[index];
307             while(tmp -> next != NULL){
308                 if(strcmp(tmp->opcode, node->opcode) == 0){   //有重複
309                     flag = 1;
310                     break;
311                 }else{
312                     tmp = tmp -> next;
313                 }
314             }
315             if(strcmp(tmp->opcode, node->opcode) != 0){
316                 tmp -> next = node;
317             }else{
318                 flag = 1;
319             }
320         }
321
322         if(flag == 0){
323             if(lit_head == NULL){
324                 lit_head = temp;
325             }else{
326                 list t = lit_head;
327                 while(t -> next != NULL){
328                     t = t -> next;
329                 }
330                 t -> next = temp;
331             }
332         }
333 }
```

```
334 void buildSymTab(int index,list node){
335         list ptr = newnode();
336         ptr -> address = node -> address;
337         ptr -> block = node -> block;
338         strcpy(ptr->name, node->name);
339 
340         if(symTab[index] == NULL){
341                 symTab[index] = newnode();
342                 symTab[index] = ptr;
343         }else{
344                 list tmp = symTab[index];
345                 while(tmp->next != NULL){
346                         tmp = tmp -> next;
347                 }
348                 tmp -> next = ptr;
349         }
350 }
351 Use buildBlock(char* str){
352         Use u = newBlock();
353         strcpy(u->name, str);
354         u -> num = use_num++;
355         if(useHead == NULL){
356                 useHead = u;
357         }else{
358                 Use tmp = useHead;
359                 while(tmp->next != NULL){
360                         tmp = tmp -> next;
361                 }
362                 tmp -> next = u;
363         }
364         return u;
365 }
366 void clearList(Use use){
367         int index;
368         list ptr;
369         while(lit_head != NULL){
370                 index = Hash(lit_head->opcode);
371                 ptr = searchLitTab(litTab[index], lit_head->opcode);
372                 lit_head -> address = use -> size;
373                 ptr -> address = use -> size;
374                 ptr -> block = use;
375                 lit_head -> block = use;
```

```
376                 if(lit_head -> opcode[0] == 'C'){
377                         use -> size += strlen(lit_head -> opcode) - 3;
378                 }else if(lit_head -> opcode[0] == 'X'){
379                         use -> size += (strlen(lit_head -> opcode) - 3) / 2;
380                 }
381                 lit_head = lit_head -> next;
382         }
383         lit_head = NULL;
384 }
385 list searchLitTab(list ptr, char* str){
386         while(ptr != NULL){
387                 if(!strcmp(ptr->opcode, str)){
388                         return ptr;
389                 }else{
390                         ptr = ptr -> next;
391                 }
392         }
393         return NULL;
394 }
395 list searchSymTab(list ptr, char* str){
396         while(ptr != NULL){
397                 if(!strcmp(ptr->name, str)){                //find
398                         return ptr;
399                 }else{
400                         ptr = ptr -> next;
401                 }
402         }
403         return NULL;
404 }
405 Use searchBlock(char* str){
406         Use tmp = useHead;
407         while(tmp != NULL){
408                 if(!strcmp(tmp->name, str)){
409                         return tmp;
410                 }else{
411                         tmp = tmp -> next;
412                 }
413         }
414         return NULL;
415 }
416 int searchOpTab(char* str){
417         int i;
```

```
418         for(i=0; i<20; i++){
419             if(!strcmp(optab[i].name, str)){
420                 return i;
421             }
422         }
423         return -1;
424 }
425 int searchReg(char* str){
426         int i;
427         for(i=0; i<9; i++){
428             if(!strcmp(regtab[i].name, str)){
429                 return regtab[i].num;
430             }
431         }
432         return -1;
433 }
434 void onepass(char* fname){      //建 symTab 、litTab 、address
435         sprintf(Fname,"%s",fname);
436         char c;
437         char str[100];
438         int flag, index=0, tabIndex=0;
439         memset(str,'\0',100);
440         useHead = newBlock();
441         useHead -> num = 0;
442         strcpy(useHead->name, "DEFAULT");
443         Use use = useHead;
444
445         FILE *fp = fopen(fname,"r");      //讀取檔案
446         while(1){
447             flag = fscanf(fp,"%c",&c);
448             if(c != '\0' && c != '\n' && flag != EOF){
449                 str[index] = c;
450                 index++;
451             }else{
452                 str[index] = '\0';
453                 index = 0;
454                 if(str[0] == '.') continue;            //註解跳過
455                 list node = setnode(str);    //把 str 的內容分類 ，並串起來
456                 if(!strcmp(node->opcode, "USE")){    //USE 分 BLOCK
457                     if(!strcmp(node->oper1, "\0")){ //第一區
458                         use = useHead;
459                     }else{
```

```
460                     use =    searchBlock(node->oper1);
461                     if(use == NULL){//沒有定義 block 的就創一個
462                         use = buildBlock(node->oper1);
463                     }
464                 }
465             }
466             node -> block = use;
467             node -> address = use -> size;
468             strcpy(node->target, "\0");
469
470             if(node->mark == '='){                  //literal
471                 tabIndex = Hash(node->oper1);
472                 buildLitTab(tabIndex, node, use);
473             }
474             if(!strcmp(node->opcode, "LTORG")){ //LTORG  常數要加在下面一行
475                 node -> next = lit_head;
476                 clearList(use);
477             }
478             if(!strcmp(node->opcode, "END")){   //END  常數要加在下面一行
479                 node -> next = lit_head;
480                 clearList(use);
481             }
482             if(!strcmp(node->opcode, "EQU")){   //EQU  要做運算 + - * /之外的不做運算
483                 tabIndex = Hash(node->oper1);
484                 list op1 = searchSymTab(symTab[tabIndex], node->oper1);
485                 tabIndex = Hash(node->oper2);
486                 list op2 = searchSymTab(symTab[tabIndex], node->oper2);
487                 if(node->oper == '+'){
488                     node->address = op1->address + op2->address;
489                 }else if(node->oper == '-'){
490                     node->address = op1->address - op2->address;
491                 }else if(node->oper == '*'){
492                     node->address = op1->address * op2->address;
493                 }else if(node->oper == '/'){
494                     node->address = op1->address / op2->address;
495                 }
496             }
497             //課本 2-11
498             if(!strcmp(node->opcode, "START")){
499                 node -> address = atoi(node->oper1);
500                 use -> address = atoi(node->oper1);
501                 use -> size = atoi(node->oper1);
```

```
502                     }else{
503                             int len = 0;
504                             int num = searchOpTab(node->opcode);
505                             if(num != -1){
506                                     if(optab[num].format[0] == '2'){
507                                             len = 2;
508                                     }else if(node->extend == '+'){
509                                             len = 4;
510                                     }else{
511                                             len = 3;
512                                     }
513                             }else if(!strcmp(node->opcode, "WORD")){
514                                     len = 3;
515                             }else if(!strcmp(node->opcode, "RESW")){
516                                     len = 3 * atoi(node->oper1);
517                             }else if(!strcmp(node->opcode, "RESB")){
518                                     len = atoi(node->oper1);
519                             }else if(!strcmp(node->opcode, "BYTE")){
520                                     len = strlen(node->oper1) - 3;
521                                     if(node->oper1[0] == 'X') len /= 2;
522                             }
523                             if(node->name[0] != '\0'){  //symTab
524                                     tabIndex = Hash(node->name);
525                                     buildSymTab(tabIndex, node);
526                             }
527                             use->size += len;
528                             len = 0;
529                     }
530                     if(flag == EOF) break;
531             }
532     }
533     int cnt = useHead -> address;
534     use = useHead;
535     while(use != NULL){
536             use -> address = cnt;
537             cnt += use -> size;
538             use = use -> next;
539     }
540 }
541 void twopass(){
542     list ptr = head;
543     int index, opni, num, xbpe, disp, i;
```

```
544         list tmp;
545         char str[10],dispStr[10];
546         char objStr[80], tmpStr[80];
547         while(ptr != NULL){
548             if(!strcmp(ptr->opcode, "BASE")){
549                 index = Hash(ptr->oper1);
550                 tmp = searchSymTab(symTab[index], ptr->oper1);
551                 base = tmp -> address;
552                 strcpy(ptr->target,"\0");
553             }else{
554                 index = searchOpTab(ptr->opcode);
555                 if(index != -1){
556                     opni = (int)strtol(optab[index].code, NULL, 16);
557                     if(ptr->mark == '#'){
558                         opni += 1;
559                     }else if(ptr->mark == '@'){
560                         opni += 2;
561                     }else{
562                         opni += 3;
563                     }
564
565                     int Index;
566                     Index = Hash(ptr->oper1);
567                     if(ptr->mark == '='){          //oper1 is literal
568                         tmp = searchLitTab(litTab[Index], ptr->oper1);
569                     }else{                  //oper1 is symbol
570                         tmp = searchSymTab(symTab[Index], ptr->oper1);
571                     }
572
573                     if(ptr->extend == '+'){        //set pc
574                         pc = ptr->address + 4;
575                     }else if(optab[Index].format[0] == '3'){
576                         pc = ptr->address + 3;
577                     }else{
578                         pc = ptr->address + 2;
579                     }
580
581                     xbpe = 0;
582                     if(ptr->extend == '+'){        //格式四   x = 1 : xx1xxxxx
583                         if(tmp != NULL){
584                             sprintf(str, "%02X1%05X", opni, tmp->address);
585                         }else{
```

```
586                             sprintf(str, "%02X1%05X", opni, atoi(ptr->oper1));
587                         }
588                         strcpy(ptr->target, str);
589                     }else if(optab[index].format[0] == '3'){       //格式三
590                         if(tmp == NULL){       //not literal & symbol
591                             num = atoi(ptr->oper1);
592                             sprintf(dispStr, "%03X", num);
593                         }else{
594                             disp = tmp->address + tmp->block->address - pc;
595                             if(-2048 > disp || disp > 2047){          //base
596                                 xbpe += 4;
597                                 disp = tmp->address - base;       //TA - base
598                             }else{        //program counter
599                                 xbpe += 2;                //TA - pc
600                             }
601                             sprintf(dispStr, "%03X", disp);
602                             if(disp < 0){
603                                 sprintf(dispStr, "%s", dispStr+5);
604                             }
605                             if(!strcmp(ptr->oper2, "X")){
606                                 xbpe += 8;
607                             }
608                         }
609                         sprintf(str, "%02X%X%s",opni, xbpe, dispStr);
610                         strcpy(ptr->target, str);
611                     }else{       //格式二
612                         opni -= 3;
613                         if(!strcmp(ptr->oper2, "\0")){
614                             sprintf(str, "%X%X0", opni, searchReg(ptr->oper1));
615                         }else{
616                             sprintf(str, "%X%X%X", opni, searchReg(ptr->oper1), searchReg(ptr->oper2));
617                         }
618                         strcpy(ptr->target, str);
619                     }
620                 }else if(ptr->extend == '='){
621                     int cnt = 0;         //拿 str 去存
622                     memset(str, '\0', 10);
623                     for(i=2; i<strlen(ptr->opcode)-1; i++){
624                         if(ptr->opcode[0] == 'C'){
625                             str[cnt++] = ptr->opcode[i] / 16 + '0';
626                             if(ptr->opcode[i]%16 >= 10){
627                                 str[cnt++] = ptr->opcode[i] % 16 + '7';
```

```
628                                     }else{
629                                         str[cnt++] = ptr->opcode[i] % 16 + '0';
630                                     }
631                                 }else{
632                                     str[cnt++] = ptr->opcode[i];
633                                 }
634                             }
635                             str[cnt] == '\0';
636                             strcpy(ptr->target, str);
637                         }else if(!strcmp(ptr->opcode, "BYTE")){
638                             int cnt = 0;
639                             for(i=2; i<strlen(ptr->oper1)-1; i++){
640                                 if(ptr->oper1[0] == 'C'){
641                                     ptr->target[cnt++] = ptr->oper1[i] / 16 + '0';
642                                     if(ptr->opcode[i]%16 >= 10){
643                                         ptr->target[cnt++] = ptr->oper1[i] % 16 + '7';
644                                     }else{
645                                         ptr->target[cnt++] = ptr->oper1[i] % 16 + '0';
646                                     }
647                                 }else{
648                                     ptr->target[cnt++] = ptr->oper1[i];
649                                 }
650                             }
651                             ptr->target[cnt] == '\0';
652                         }else if(!strcmp(ptr->opcode, "WORD")){
653                             sprintf(ptr->target, "%X", atoi(ptr->oper1));
654                         }
655                     }
656                     ptr = ptr -> next;
657                 }
658 }
659 void Object_program(){
660         FILE* fp = fopen("D0713019_蔡聿涵_OBJFILE.txt","w");
661         fprintf(fp,"Filename: %s\n",Fname);
662         fprintf(fp,"----------【OBJ Program】----------\n");
663         int count = 0;
664         list Hptr = newnode();
665         Use use = useHead;
666         // H
667         while(use != NULL){
668             count += use -> size;
669             use = use -> next;
```

```
670         }
671         sprintf(Hptr->str, "H%-06s%06X%06X", head->name, head->address, count);
672         fprintf(fp, "%s\n",Hptr->str);
673         // T
674         list ptr = head;
675         list mHead = NULL;
676         list first = NULL;
677         list tmp;
678         int flag = 1, index;
679         char str[80], temp[80], objStr[80];
680         memset(str, '\0', 80);
681         count = 0;
682         while(ptr != NULL){
683                 if(strlen(ptr->target) != 0){
684                         if(flag == 1){
685                                 first = ptr;
686                                 flag = 0;
687                         }
688                         if(ptr->extend == '+' && ptr->mark != '#'){  // M
689                                 index = Hash(ptr->oper1);
690                                 tmp = searchSymTab(symTab[index], ptr->oper1);
691                                 if(tmp){
692                                         sprintf(temp, "M%06X05", ptr->address+1);
693                                         list nwptr = newnode();
694                                         strcpy(nwptr->str, temp);
695                                         list node = mHead;
696                                         if(mHead == NULL){
697                                                 mHead = nwptr;
698                                         }else{
699                                                 while(node->next != NULL){
700                                                         node = node -> next;
701                                                 }
702                                                 node -> next = nwptr;
703                                         }
704                                 }
705                         }
706                         strcat(str, ptr->target);
707                         count += strlen(ptr->target) / 2;
708
709                         if(useHead->next != NULL){
710                                 if(ptr->block->num != ptr->next->block->num || strcmp(ptr->next->opcode, "END") == 0){
711                                         flag = 1;
```

```
712                         }
713                 }
714                 if(ptr -> next == NULL){
715                         flag = 1;
716                 }else{
717                         if(!strcmp(ptr->next->opcode, "RESW")){
718                                 flag = 1;
719                         }
720                         if(!strcmp(ptr->next->opcode, "RESB")){
721                                 flag = 1;
722                         }
723                 }
724                 if(count >= 29 || flag == 1){
725                         sprintf(temp,"T%06X%02X",first->address+first->block->address, count);
726                         sprintf(objStr, "%s%s",temp,str);
727                         fprintf(fp, "%s\n",objStr);
728                         memset(str, '\0', strlen(str));
729                         memset(temp, '\0',strlen(temp));
730                         memset(temp, '\0',strlen(objStr));
731                         count = 0;
732                         flag = 1;
733                 }
734         }
735         ptr = ptr -> next;
736 }
737 if(strlen(str) != 0){
738         sprintf(temp, "T%06X%02X", first->address+first->block->address, count);
739         sprintf(objStr, "%s%s",temp,str);
740         fprintf(fp,"%s\n",objStr);
741         memset(str, '\0', strlen(str));
742         memset(temp, '\0',strlen(temp));
743         memset(temp, '\0',strlen(objStr));
744 }
745 ptr = mHead;
746 while(ptr != NULL){
747         fprintf(fp, "%s\n",ptr->str);
748         ptr = ptr -> next;
749 }
750 //E
751 ptr = head;
752 while(ptr != NULL){
753         index = searchOpTab(ptr->opcode);
```

```
754             if(index != -1){
755                 sprintf(temp, "E%06X", ptr->address);
756                 break;
757             }
758             ptr = ptr->next;
759         }
760         fprintf(fp, "%s\n", temp);
761 }
762 int main(){
763 //      onepass("srcpro2.9.txt");
764         onepass("D0713019_蔡聿涵_srcpro.txt");
765         twopass();
766         Object_program();
767         printPool();
768         printLitTab();
769         printSymTab();
770         printOpTab();
771         printRegTab();
772 }
```

二、目的程式碼

```
Filename: srcpro2.9.txt
---------------【OBJ Program】---------------
HCOPY  000000001077
T0000001D17202D69202D4B10103603202629000033200074B10105D3F2FEC032010
T00001D130F20160100030F200D4B10105D3E2003454F46
T0010361DB410B400B44075101000E32019332FFADB2013A00433200857C003B850
T0010531D3B2FEA1340004F0000F1B410774000E32011332FFA53C003DF2008B850
T001070073B2FEF4F000005
M00000705
M00001405
M00002705
E000000
```

```
Filename: srcpro2.11.txt
---------------【OBJ Program】---------------
HCOPY  000000001071
T0000001E1720634B20210320602900003320064B203B3F2FEE0320550F2056010003
T00001E090F20484B20293E203F
T0000271DB410B400B44075101000E32038332FFADB2032A00433200857A02FB850
T000044093B2FEA13201F4F0000
T00006C01F1
T00004D19B410772017E3201B332FFA53A016DF2012B8503B2FEF4F0000
T00006D04454F4605
E000000
```

三、Fig.2.9 執行結果

1. 組合語言原始敘述

```
Filename: srcpro2.9.txt
----------------------------【Literal Pool】----------------------------
Row Address Block       Code                                    Target

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. LITTAB

```
Filename: srcpro2.9.txt
--------------【LITTAB】---------------
 Row    LitName   Address    UseName
   1     C'EOF'   002D       DEFAULT
   2      X'05'   1076       DEFAULT
```

3. SYMTAB

```
Filename: srcpro2.9.txt
--------------【SYMTAB】---------------
 Row    SymName   Address    UseName
   1      RLOOP   1040       DEFAULT
   2     BUFFER   0036       DEFAULT
   3     MAXLEN   1000       DEFAULT
   4      WRREC   105D       DEFAULT
   5      INPUT   105C       DEFAULT
   6     ENDFIL   001A       DEFAULT
   7      RDREC   1036       DEFAULT
   8      WLOOP   1062       DEFAULT
   9       EXIT   1056       DEFAULT
  10      FIRST   0000       DEFAULT
  11      CLOOP   0006       DEFAULT
  12     BUFEND   1036       DEFAULT
  13     RETADR   0030       DEFAULT
  14     LENGTH   0033       DEFAULT
```

4. OPTAB

```
Filename: srcpro2.9.txt
--------------【OPTAB】---------------
 Row   OpName     Format   OpCode   Infor
   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     LDCH      3/4      50       m
  17       WD      3/4      DC       m
  18      JLT      3/4      38       m
  19      STX      3/4      10       m
  20     RSUB      3/4      4C       NULL
```

5. REGTAB

```
Filename: srcpro2.9.txt
--------------【REGTAB】---------------
 Row   RegName   RegCode
   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
```

四、Fig.2.11 執行結果

1. 組合語言原始敘述

```
Filename: srcpro2.11.txt
---------------------------- 【Literal Pool】 ----------------------------
Row Address Block       Code                                    Target

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      TX                  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. LITTAB

```
Filename: srcpro2.11.txt
-------------- 【LITTAB】 --------------
 Row   LitName  Address   UseName
   1    C'EOF'  0007      CDATA
   2     X'05'  000A      CDATA
```

3. SYMTAB

```
Filename: srcpro2.11.txt
-------------- 【SYMTAB】 --------------
 Row   SymName  Address   UseName
   1     RLOOP  0031      DEFAULT
   2    BUFFER  0000      CBLKS
   3    MAXLEN  1000      CBLKS
   4     WRREC  004D      DEFAULT
   5     INPUT  0006      CDATA
   6    ENDFIL  0015      DEFAULT
   7     RDREC  0027      DEFAULT
   8     WLOOP  0052      DEFAULT
   9      EXIT  0047      DEFAULT
  10     FIRST  0000      DEFAULT
  11     CLOOP  0003      DEFAULT
  12    BUFEND  1000      CBLKS
  13    RETADR  0000      CDATA
  14    LENGTH  0003      CDATA
```

4. OPTAB

```
Filename: srcpro2.11.txt
-------------- 【OPTAB】 --------------
 Row  OpName    Format  OpCode  Infor
   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    LDCH    3/4     50      m
  17      WD    3/4     DC      m
  18     JLT    3/4     38      m
  19     STX    3/4     10      m
  20    RSUB    3/4     4C      NULL
```

5. REGTAB

```
Filename: srcpro2.11.txt
-------------- 【REGTAB】 --------------
 Row  RegName   RegCode
   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
```