

系統程式期末報告

SIC simulator

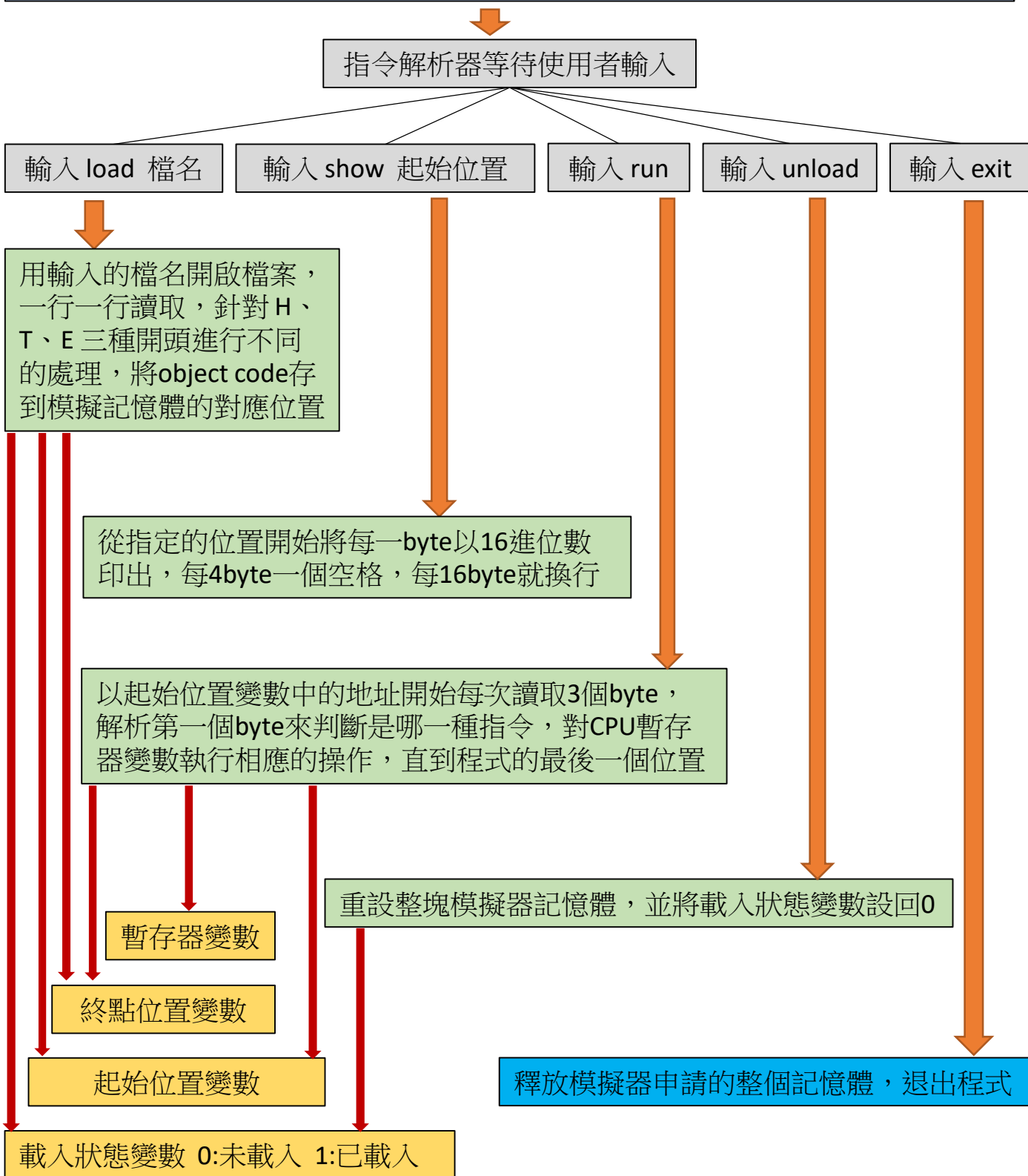
李奕承 611121212

目標概述

這個作業的目標是要做出一個模擬器，他可以讀取object code並載入到記憶體中，還可以dump記憶體中的內容，以及模擬執行object code。

運作原理

剛開始執行時，先申請一大塊記憶體來模擬一台電腦
建立一個結構來代表CPU，成員是 SIC 的五個暫存器(用 24bit 的無號整數來模擬)



```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 #include"tool.h"
5
6
7 typedef unsigned char uint8_t;
8
9
10 typedef struct cpu_t{
11     unsigned int A : 24;
12     unsigned int X : 24;
13     unsigned int L : 24;
14     unsigned int PC : 24;
15     unsigned int SW : 24;
16 }cpu_t;
17
18 void show(void * memory, char * begin, char * end);
19
20 void load(char * target, void * memory, int * program_begin, int * program_end, int
* loaded);
21
22 void run(int * program_begin, int * program_end, void * memory, cpu_t * cpu);
23
24
25
26
27
```

```
1 #ifndef __TOOL_H_
2 #define __TOOL_H_
3
4 #include<stdio.h>
5 #include<stdlib.h>
6 #include<string.h>
7
8 int exponent_Int(const int base, int n);
9
10 int hex_to_dex(char *hex);
11
12 int hex_to_dex_c(char hex);
13
14 #endif
```

```

1 #include "simulator.h"
2
3
4
5 int main(int argc, char *argv[]){
6     char input[100];
7     char temp1[100];
8     char temp2[100];
9     void * memory = NULL;
10    int system_size = 0;
11    int loaded = 0;
12    int program_begin = 0;
13    int program_end = 0;
14    if(argv[1] == 0)
15    {
16        memory = malloc(hex_to_dex("5000"));
17        memset(memory, '.', hex_to_dex("5000"));
18        system_size = hex_to_dex("5000");
19        printf("system memory from 0x0 ~ 0x%X\n", system_size);
20    }
21    else
22    {
23        system_size = hex_to_dex(argv[1]);
24        printf("system memory from 0x0 ~ 0x%X\n", system_size);
25        memory = malloc(system_size);
26        memset(memory, '.', system_size);
27    }
28
29
30    temp2[0] = '\0';
31    temp2[1] = '\0';
32    temp1[0] = '\0';
33
34    //CPU
35    cpu_t cpu;
36    cpu.A = 0;
37    cpu.X = 0;
38    cpu.L = 0;
39    cpu.PC = 0;
40    cpu.SW = 0;
41
42
43
44
45    while(1)
46    {
47        printf(">>> ");
48        fgets(input, 100, stdin);
49        sscanf(input, "%s %s", temp1, temp2);
50
51        if(strcmp(temp1, "exit") == 0)
52        {
53            free(memory);
54            break;
55        }
56
57        else if(strcmp(temp1, "show") == 0)
58        {
59            show(memory, temp2, argv[1]);
60            temp2[0] = '\0';
61            temp2[1] = '\0';
62            temp1[0] = '\0';
63        }
64
65        else if(strcmp(temp1, "load") == 0)
66        {
67            if(loaded == 0)
68            {
69                load(temp2, memory, &program_begin, &program_end, &loaded);
70            }
71            else
72            {
73                printf("error , an object code has been load\n");
74            }
75
76
77            temp2[0] = '\0';
78            temp2[1] = '\0';

```

```

79         temp1[0] = '\0';
80     }
81
82     else if(strcmp(temp1, "unload") == 0)
83     {
84         memset(memory, '.', system_size);
85         cpu.A = 0;
86         cpu.X = 0;
87         cpu.L = 0;
88         cpu.PC = 0;
89         cpu.SW = 0;
90         loaded = 0;
91         temp2[0] = '\0';
92         temp2[1] = '\0';
93         temp1[0] = '\0';
94         printf("A : %06X  X : %06X  L : %06X  PC : %06X  SW : %06X\n\n", cpu.A,
cpu.X, cpu.L, cpu.PC, cpu.SW);
95     }
96
97     else if(strcmp(temp1, "run") == 0)
98     {
99         run(&program_begin, &program_end, memory, &cpu);
100         temp2[0] = '\0';
101         temp2[1] = '\0';
102         temp1[0] = '\0';
103         printf("A : %06X  X : %06X  L : %06X  PC : %06X  SW : %06X\n\n", cpu.A,
cpu.X, cpu.L, cpu.PC, cpu.SW);
104     }
105
106     else
107     {
108         printf("command not find\n");
109     }
110 }
111 }
112
113
114

```

```

1 #include "simulator.h"
2
3 void load(char * target, void * memory, int * program_begin, int * program_end, int
* loaded)
4 {
5     char buffer[100];
6     FILE * load_obj = fopen(target, "r");
7     if(load_obj == NULL)
8     {
9         printf("error\n");
10    }
11    else
12    {
13        *loaded = 1;
14        char begin[20];
15        int size = 0;
16        while(fgets(buffer, 100, load_obj))
17        {
18            if(*buffer == 'H')
19            {
20                for(int i = 0 ; i < 6 ; i++)
21                {
22                    begin[i] = buffer[i + 7];
23                }
24                begin[6] = '\0';
25                printf("%d ", hex_to_dex(begin));
26                *program_begin = hex_to_dex(begin);
27
28                size = hex_to_dex(buffer+13);
29                printf("%d \n", size);
30                *program_end = *program_begin + size;
31            }
32
33            if(*buffer == 'T')
34            {
35                for(int i = 0 ; i < 6 ; i++)
36                {
37                    begin[i] = buffer[i + 1];
38                }
39                begin[6] = '\0';
40
41                printf("%X ", hex_to_dex(begin));
42                int memory_count = hex_to_dex(begin);
43                int state = 0;
44
45                for(int i = 9 ; i < strlen(buffer) - 1 ; i++)
46                {
47                    if(state == 0)
48                    {
49                        *((uint8_t *)memory + memory_count) =
50(hex_to_dex_c(buffer[i]) << 4) ;
51                        printf("%c", buffer[i]);
52                        state = 1;
53                    }
54                    else if(state == 1)
55                    {
56                        *((uint8_t *)memory + memory_count) |=
57(hex_to_dex_c(buffer[i]) << 0) ;
58                        memory_count++;
59                        printf("%c", buffer[i]);
60                        state = 0;
61                    }
62                }
63                printf("\n");
64            }
65        }
66        fclose(load_obj);
67    }
68 }

```

```

1 #include "simulator.h"
2
3
4 void show(void * memory, char * begin, char * end)
5 {
6
7     int offset = 0;
8     if(hex_to_dex(begin) <= hex_to_dex(end)-1)
9     {
10         if(hex_to_dex(begin) % 16 != 0)
11         {
12             printf("%04X ", hex_to_dex(begin));
13             for(int i = 0 ; i < (hex_to_dex(begin) % 16) ; i++)
14             {
15                 printf(" ");
16                 offset++;
17                 if(offset == 4 || offset == 8 || offset == 12)
18                 {
19                     printf(" ");
20                 }
21             }
22         }
23
24         for(int i = hex_to_dex(begin); i < hex_to_dex(end); i++)
25         {
26             if(i % 16 == 0)
27             {
28                 offset = 0;
29                 printf("%04X ", i);
30             }
31
32             if(*((char *) (memory+i)) == '.')
33             {
34                 printf("..");
35             }
36             else
37             {
38                 printf("%02X", *((uint8_t *) (memory+i)));
39             }
40
41             offset++;
42
43             if(offset == 4 || offset == 8 || offset == 12)
44             {
45                 printf(" ");
46             }
47
48             if(offset == 16)
49             {
50                 printf("\n");
51             }
52         }
53         offset = 0;
54         printf("\n");
55     }
56     else
57     {
58         printf("error %d\n", hex_to_dex(begin));
59     }
60 }

```



```

1 #include "simulator.h"
2
3
4
5 void run(int * program_begin, int * program_end, void * memory, cpu_t * cpu)
6 {
7     printf("%X %X\n", *program_begin, *program_end);
8     cpu->PC = *program_begin;
9     int temp = 0;
10    int temp2 = 0;
11    int temp3 = 0;
12    int x_mod = 0;
13    FILE * input = NULL;
14    FILE * output = NULL;
15    input = fopen("F1.txt", "r");
16    output = fopen("F5.txt", "w");
17    cpu->L = *program_end;
18
19
20    while(cpu->PC < *program_end)
21    {
22        x_mod = 0;
23        temp = 0;
24        temp2 = 0;
25        temp3 = 0;
26        if(*((uint8_t *)memory + cpu->PC + 1) >= 128)
27        {
28            x_mod = 1;
29        }
30
31        switch(*((uint8_t *)memory + cpu->PC))
32        {
33            case 24:
34                printf("ADD      ");
35                temp = *((uint8_t *)memory + cpu->PC + 1);
36                temp = temp << 8;
37                temp |= *((uint8_t *)memory + cpu->PC + 2);
38
39                temp2 = *((uint8_t *)memory + temp + 2);
40                cpu->A += temp2;
41
42                temp2 = *((uint8_t *)memory + temp + 1);
43                temp2 = temp2 << 8;
44                cpu->A += temp2;
45
46                temp2 = *((uint8_t *)memory + temp);
47                temp2 = temp2 << 16;
48                cpu->A += temp2;
49
50
51                cpu->PC += 3;
52                break;
53
54            case 52:
55                printf("JGT      ");
56                cpu->PC += 3;
57                break;
58
59            case 0:
60                printf("LDA      ");
61                temp = *((uint8_t *)memory + cpu->PC + 1);
62                temp = temp << 8;
63                temp |= *((uint8_t *)memory + cpu->PC + 2);
64
65                if(*((uint8_t *)memory + temp) == '.')
66                {
67                    cpu->A = 0;
68                }
69                else
70                {
71                    cpu->A = *((uint8_t *)memory + temp);
72                }
73
74                cpu->A = cpu->A << 8;
75
76                if(*((uint8_t *)memory + temp + 1) == '.')
77                {
78                    cpu->A |= 0;

```

```

79     }
80     else
81     {
82         cpu->A |= *((uint8_t *)memory + temp + 1);
83     }
84
85     cpu->A = cpu->A << 8;
86
87     if(*((uint8_t *)memory + temp + 2) == '.')
88     {
89         cpu->A |= 0;
90     }
91     else
92     {
93         cpu->A |= *((uint8_t *)memory + temp + 2);
94     }
95
96
97
98     cpu->PC += 3;
99     break;
100
101 case 4:
102     printf("LDX      ");
103     temp = *((uint8_t *)memory + cpu->PC + 1);
104     temp = temp << 8;
105     temp |= *((uint8_t *)memory + cpu->PC + 2);
106
107     if(*((uint8_t *)memory + temp) == '.')
108     {
109         cpu->X = 0;
110     }
111     else
112     {
113         cpu->X = *((uint8_t *)memory + temp);
114     }
115
116     cpu->X = cpu->X << 8;
117
118     if(*((uint8_t *)memory + temp + 1) == '.')
119     {
120         cpu->X |= 0;
121     }
122     else
123     {
124         cpu->X |= *((uint8_t *)memory + temp + 1);
125     }
126
127     cpu->X = cpu->X << 8;
128
129     if(*((uint8_t *)memory + temp + 2) == '.')
130     {
131         cpu->X |= 0;
132     }
133     else
134     {
135         cpu->X |= *((uint8_t *)memory + temp + 2);
136     }
137
138
139     cpu->PC += 3;
140     break;
141
142 case 216:
143     printf("RD      ");
144     temp = *((uint8_t *)memory + cpu->PC + 1);
145     temp = temp << 8;
146     temp |= *((uint8_t *)memory + cpu->PC + 2);
147
148
149     //cpu->A = *((uint8_t *)memory + temp);
150
151     temp2 = getc(input);
152     if(temp2 == -1)
153     {
154         cpu->A = 0;
155     }
156     else
157     {

```

```

158         cpu->A = (unsigned int)temp2;
159     }
160
161
162
163     cpu->PC += 3;
164     break;
165
166 case 220:
167     printf("WD      ");
168     temp = *((uint8_t *)memory + cpu->PC + 1);
169     temp = temp << 8;
170     temp |= *((uint8_t *)memory + cpu->PC + 2);
171
172     /*((uint8_t *)memory + temp) = cpu->A;
173     putc(cpu->A, output);
174
175     cpu->PC += 3;
176     break;
177
178 case 84:
179     printf("STCH    ");
180     temp = *((uint8_t *)memory + cpu->PC + 1);
181     temp = temp << 8;
182     temp |= *((uint8_t *)memory + cpu->PC + 2);
183
184     if(x_mod == 1)
185     {
186         temp = temp ^ (1 << 15);
187         *((uint8_t *)memory + temp + cpu->X) = cpu->A;
188     }
189     else
190     {
191         *((uint8_t *)memory + temp) = cpu->A;
192     }
193
194     cpu->PC += 3;
195     break;
196
197 case 16:
198     printf("STX      ");
199     temp = *((uint8_t *)memory + cpu->PC + 1);
200     temp = temp << 8;
201     temp |= *((uint8_t *)memory + cpu->PC + 2);
202
203     temp2 = cpu->X;
204
205     *((uint8_t *)memory + temp + 2) = temp2;
206     temp2 = temp2 >> 8;
207     *((uint8_t *)memory + temp + 1) = temp2;
208     temp2 = temp2 >> 8;
209     *((uint8_t *)memory + temp) = temp2;
210
211
212     cpu->PC += 3;
213
214     break;
215
216 case 12:
217     printf("STA      ");
218     temp = *((uint8_t *)memory + cpu->PC + 1);
219     temp = temp << 8;
220     temp |= *((uint8_t *)memory + cpu->PC + 2);
221
222     temp2 = cpu->A;
223
224     *((uint8_t *)memory + temp + 2) = temp2;
225     temp2 = temp2 >> 8;
226     *((uint8_t *)memory + temp + 1) = temp2;
227     temp2 = temp2 >> 8;
228     *((uint8_t *)memory + temp) = temp2;
229
230
231     cpu->PC += 3;
232
233     break;
234
235 case 44:
236     printf("TIX      ");

```

```

237     cpu->X++;
238
239     temp = *((uint8_t *)memory + cpu->PC + 1);
240     temp = temp << 8;
241     temp |= *((uint8_t *)memory + cpu->PC + 2);
242
243     temp2 = *((uint8_t *)memory + temp);
244     temp2 = temp2 << 8;
245     temp2 |= *((uint8_t *)memory + temp + 1);
246     temp2 = temp2 << 8;
247     temp2 |= *((uint8_t *)memory + temp + 2);
248
249     if(cpu->X == temp2)
250     {
251         cpu->SW = '=';
252     }
253     else if(cpu->X < temp2)
254     {
255         cpu->SW = '<';
256     }
257     else
258     {
259         cpu->SW = '>';
260     }
261
262     cpu->PC += 3;
263     break;
264
265 case 64:
266     printf("AND      ");
267     cpu->PC += 3;
268     break;
269
270 case 60:
271     printf("J      ");
272     temp = cpu->PC;
273     cpu->PC = *((uint8_t *)memory + temp + 1);
274     cpu->PC = cpu->PC << 8;
275     cpu->PC |= *((uint8_t *)memory + temp + 2);
276     break;
277
278 case 56:
279     printf("JLT      ");
280     temp = cpu->PC;
281     if(cpu->SW == '<')
282     {
283         cpu->PC = *((uint8_t *)memory + temp + 1);
284         cpu->PC = cpu->PC << 8;
285         cpu->PC |= *((uint8_t *)memory + temp + 2);
286     }
287     else
288     {
289         cpu->PC += 3;
290     }
291     break;
292
293 case 80:
294     printf("LDCH     ");
295     temp = *((uint8_t *)memory + cpu->PC + 1);
296     temp = temp << 8;
297     temp |= *((uint8_t *)memory + cpu->PC + 2);
298
299     if(x_mod == 1)
300     {
301         temp = temp ^ (1 << 15);
302         cpu->A = *((uint8_t *)memory + temp + cpu->X);
303     }
304     else
305     {
306         cpu->A = *((uint8_t *)memory + temp);
307     }
308
309     cpu->PC += 3;
310     break;
311
312 case 32:
313     printf("MUL      ");
314

```

```

316     temp = *((uint8_t *)memory + cpu->PC + 1);
317     temp = temp << 8;
318     temp |= *((uint8_t *)memory + cpu->PC + 2);
319
320     temp3 = cpu->A;
321     cpu->A = 0;
322
323     temp2 = *((uint8_t *)memory + temp + 2);
324     cpu->A += temp3 * temp2;
325
326     temp2 = *((uint8_t *)memory + temp + 1);
327     temp2 = temp2 << 8;
328     cpu->A += temp3 * temp2;
329
330     temp2 = *((uint8_t *)memory + temp);
331     temp2 = temp2 << 16;
332     cpu->A += temp3 * temp2;
333
334
335     cpu->PC += 3;
336     break;
337
338 case 76:
339     printf("RSUB    ");
340     cpu->PC = cpu->L;
341     break;
342
343 case 20:
344     printf("STL      ");
345     temp = *((uint8_t *)memory + cpu->PC + 1);
346     temp = temp << 8;
347     temp |= *((uint8_t *)memory + cpu->PC + 2);
348
349     temp2 = cpu->L;
350
351     *((uint8_t *)memory + temp + 2) = temp2;
352     temp2 = temp2 >> 8;
353     *((uint8_t *)memory + temp + 1) = temp2;
354     temp2 = temp2 >> 8;
355     *((uint8_t *)memory + temp) = temp2;
356
357
358
359     cpu->PC += 3;
360     break;
361
362 case 28:
363     printf("SUB      ");
364     temp = *((uint8_t *)memory + cpu->PC + 1);
365     temp = temp << 8;
366     temp |= *((uint8_t *)memory + cpu->PC + 2);
367
368     temp2 = *((uint8_t *)memory + temp);
369     temp2 = temp2 << 8;
370
371     temp2 = *((uint8_t *)memory + temp + 1);
372     temp2 = temp2 << 8;
373
374     temp2 = *((uint8_t *)memory + temp + 2);
375
376     cpu->A = cpu->A - temp2;
377
378
379     cpu->PC += 3;
380     break;
381
382
383
384 case 40:
385     printf("COMP     ");
386     temp = *((uint8_t *)memory + cpu->PC + 1);
387     temp = temp << 8;
388     temp |= *((uint8_t *)memory + cpu->PC + 2);
389
390     temp2 = *((uint8_t *)memory + temp);
391     temp2 = temp2 << 8;
392     temp2 |= *((uint8_t *)memory + temp + 1);
393     temp2 = temp2 << 8;
394     temp2 |= *((uint8_t *)memory + temp + 2);

```

```

395
396     if(cpu->A == temp2)
397     {
398         cpu->SW = '=';
399     }
400     else if(cpu->A < temp2)
401     {
402         cpu->SW = '<';
403     }
404     else
405     {
406         cpu->SW = '>';
407     }
408
409     cpu->PC += 3;
410     break;
411
412 case 48:
413     printf("JEQ      ");
414     temp = cpu->PC;
415     if(cpu->SW == '=')
416     {
417         cpu->PC = *((uint8_t *)memory + temp + 1);
418         cpu->PC = cpu->PC << 8;
419         cpu->PC |= *((uint8_t *)memory + temp + 2);
420
421     }
422     else
423     {
424         cpu->PC += 3;
425     }
426     break;
427
428 case 72:
429     printf("JSUB     ");
430     temp = cpu->PC;
431     cpu->L = cpu->PC + 3;
432     cpu->PC = *((uint8_t *)memory + temp + 1);
433     cpu->PC = cpu->PC << 8;
434     cpu->PC |= *((uint8_t *)memory + temp + 2);
435     break;
436
437 case 8:
438     printf("LDL      ");
439     temp = *((uint8_t *)memory + cpu->PC + 1);
440     temp = temp << 8;
441     temp |= *((uint8_t *)memory + cpu->PC + 2);
442
443     if(*((uint8_t *)memory + temp) == '.')
444     {
445         cpu->L = 0;
446     }
447     else
448     {
449         cpu->L = *((uint8_t *)memory + temp);
450     }
451
452     cpu->L = cpu->L << 8;
453
454     if(*((uint8_t *)memory + temp + 1) == '.')
455     {
456         cpu->L |= 0;
457     }
458     else
459     {
460         cpu->L |= *((uint8_t *)memory + temp + 1);
461     }
462
463     cpu->L = cpu->L << 8;
464
465     if(*((uint8_t *)memory + temp + 2) == '.')
466     {
467         cpu->L |= 0;
468     }
469     else
470     {
471         cpu->L |= *((uint8_t *)memory + temp + 2);
472     }
473

```

```

474
475
476     cpu->PC += 3;
477     break;
478
479 case 68:
480     printf("OR      ");
481     cpu->PC += 3;
482     break;
483
484 case 232:
485     printf("STSW    ");
486     cpu->PC += 3;
487     break;
488
489 case 224:
490     printf("TD      ");
491     temp = *((uint8_t *)memory + cpu->PC + 1);
492     temp = temp << 8;
493     temp |= *((uint8_t *)memory + cpu->PC + 2);
494
495
496     temp2 = *((uint8_t *)memory + temp);
497     //printf("%02X\n", temp2);
498
499     if(temp2 == hex_to_dex("F1"))
500     {
501         if(input != NULL)
502         {
503             cpu->SW = '<';
504         }
505         else
506         {
507             cpu->SW = '=';
508         }
509     }
510
511     if(temp2 == hex_to_dex("05"))
512     {
513         if(output != NULL)
514         {
515             cpu->SW = '<';
516         }
517         else
518         {
519             cpu->SW = '=';
520         }
521     }
522
523     /*
524     if(*((uint8_t *)memory + temp) == '.')
525     {
526         cpu->SW = '=';
527     }
528     else
529     {
530         cpu->SW = '<';
531     }
532     */
533
534     cpu->PC += 3;
535     break;
536
537 default:
538     break;
539 }
540 printf("A : %06X  X : %06X  L : %06X  PC : %06X  SW : %06X\n\n", cpu->A, cpu->X,
cpu->L, cpu->PC, cpu->SW);
541 }
542 fclose(input);
543 fclose(output);
544
545 printf("\n");
546 }
547

```

```

1 #include"tool.h"
2
3
4 //將指數乘開的函數
5 int exponent_Int(const int base, int n)
6 {
7     int p = base;
8     if(n == 0)
9     {
10         p = 1;
11     }
12     else
13     {
14         for (int i = 1; i < n; i++)
15         {
16             p *= base;
17         }
18     }
19     return p;
20 }
21
22
23 //將讀取的字串視為16進位，並轉換為10進位數字的函數
24 int hex_to_dex(char *hex)
25 {
26     char * char_temp = (char *)malloc(strlen(hex));
27     char * temp_begin = char_temp;
28     strcpy(char_temp, hex);
29     char_temp = strtok(char_temp, "\n");
30     char temp[2];
31     int total = 0;
32     int count = strlen(char_temp);
33
34     while(count-- && count >= 0)
35     {
36         sprintf(temp, "%c", *char_temp);
37         if(strcmp(temp, "A") == 0){total += exponent_Int(16, count)*10;}
38         if(strcmp(temp, "B") == 0){total += exponent_Int(16, count)*11;}
39         if(strcmp(temp, "C") == 0){total += exponent_Int(16, count)*12;}
40         if(strcmp(temp, "D") == 0){total += exponent_Int(16, count)*13;}
41         if(strcmp(temp, "E") == 0){total += exponent_Int(16, count)*14;}
42         if(strcmp(temp, "F") == 0){total += exponent_Int(16, count)*15;}
43         else{total += exponent_Int(16, count)*atoi(temp);}
44         char_temp++;
45     }
46     free(temp_begin);
47     return total;
48 }
49
50 //將讀取的字元視為16進位，並轉換為10進位數字的函數
51 int hex_to_dex_c(char hex)
52 {
53     if(hex == 'A'){return 10;}
54     if(hex == 'B'){return 11;}
55     if(hex == 'C'){return 12;}
56     if(hex == 'D'){return 13;}
57     if(hex == 'E'){return 14;}
58     if(hex == 'F'){return 15;}
59     if(hex == '9'){return 9;}
60     if(hex == '8'){return 8;}
61     if(hex == '7'){return 7;}
62     if(hex == '6'){return 6;}
63     if(hex == '5'){return 5;}
64     if(hex == '4'){return 4;}
65     if(hex == '3'){return 3;}
66     if(hex == '2'){return 2;}
67     if(hex == '1'){return 1;}
68     if(hex == '0'){return 0;}
69     return 0;
70 }

```


測試與結果

開始執行，並申請模擬器需要的空間

```
Lyciih@DESKTOP-CR5NUFU MINGW64 ~/Desktop/github/System-Programming/SIC simulator (main)
$ make run
./sic_simulator 3000
system memory from 0x0 ~ 0x3000
>>> █
```

載入test1.obj

```
Lyciih@DESKTOP-CR5NUFU MINGW64 ~/Desktop/github/System-Programming/SIC simulator (main)
$ make run
./sic_simulator 3000
system memory from 0x0 ~ 0x3000
>>> load test1.obj
4096 4218
1000 1410334820390010362810303010154820613C100300102A0C103900102D
101E 0C10364820610810334C0000454F46000003000000
2039 041030001030E0205D30203FD8205D2810303020575490392C205E38203F
2057 1010364C0000F1001000041030E02079302064509039DC20792C1036
2073 3820644C000005
>>> █
```

用show 顯示記憶體中的內容

```
>>> show
0000 .....
0010 .....
0020 .....
0030 .....
0040 .....
0050 .....
0060 .....
0070 .....
0080 .....
0090 .....
00A0 .....
```

```
0FF0 .....
1000 14103348 20390010 36281030 30101548
1010 20613C10 0300102A 0C103900 102D0C10
1020 36482061 0810334C 0000454F 46000003
1030 000000.. .....
```

```
2020 .....
2030 ..... ..041030 001030E0
2040 205D3020 3FD8205D 28103030 20575490
2050 392C205E 38203F10 10364C00 00F10010
2060 00041030 E0207930 20645090 39DC2079
2070 2C103638 20644C00 0005.... .....
```

用run模擬執行，並印出暫存器的內容

```
>>> run

LDCH  A : 000046 X : 000002 L : 001024 PC : 00206D SW : 00003C
WD    A : 000046 X : 000002 L : 001024 PC : 002070 SW : 00003C
TIX   A : 000046 X : 000003 L : 001024 PC : 002073 SW : 00003D
JLT   A : 000046 X : 000003 L : 001024 PC : 002076 SW : 00003D
RSUB  A : 000046 X : 000003 L : 001024 PC : 001024 SW : 00003D
LDL   A : 000046 X : 000003 L : 00207A PC : 001027 SW : 00003D
RSUB  A : 000046 X : 000003 L : 00207A PC : 00207A SW : 00003D

A : 000046 X : 000003 L : 00207A PC : 00207A SW : 00003D
```

用unload卸載test1.obj，重設記憶體與暫存器

```
>>> unload
A : 000000 X : 000000 L : 000000 PC : 000000 SW : 000000

>>>
```

用show看重設後的記憶體內容

```
>>> show
0000 .....
0010 .....
0020 .....

0FF0 .....
1000 .....
1010 .....
1020 .....
1030 .....
1040 .....

2000 .....
2010 .....
2020 .....
2030 .....
2040 .....
2050 .....
2060 .....
2070 .....
```

載入test2.obj

```
>>> load test2.obj
8192 45
2000 00201E1820241C201B0C20270020211820241C201B0C202A4C0000000001
201E 000005000007000003
>>> █
```

用show 顯示記憶體中的內容

2000	00201E18	20241C20	1B0C2027	00202118
2010	20241C20	1B0C202A	4C000000	00010000
2020	05000007	00000300

用run模擬執行，並印出暫存器的內容

```
>>> run
2000 202D
LDA    A : 000005 X : 000000 L : 00202D PC : 002003 SW : 000000

ADD     A : 000008 X : 000000 L : 00202D PC : 002006 SW : 000000

SUB     A : 000007 X : 000000 L : 00202D PC : 002009 SW : 000000

STA     A : 000007 X : 000000 L : 00202D PC : 00200C SW : 000000

LDA     A : 000007 X : 000000 L : 00202D PC : 00200F SW : 000000

ADD     A : 00000A X : 000000 L : 00202D PC : 002012 SW : 000000

SUB     A : 000009 X : 000000 L : 00202D PC : 002015 SW : 000000

STA     A : 000009 X : 000000 L : 00202D PC : 002018 SW : 000000

RSUB    A : 000009 X : 000000 L : 00202D PC : 00202D SW : 000000

A : 000009 X : 000000 L : 00202D PC : 00202D SW : 000000
>>> █
```

用exit退出模擬器

```
>>> exit

Lyciih@DESKTOP-CR5NUFU MINGW64 ~/Desktop/github/System-Programming/SIC simulator (main)
$ █
```

討論

在這個作業的過程中我領悟到原來市面上的很多模擬器好像都可以用這種方法來實現，只要知道該指令集的各種行為，並用變數模擬CPU中的每個暫存器，再加上模擬用的空間以及虛擬出的設備記憶體位置，就可以模擬一台電腦的運作，相當好玩。