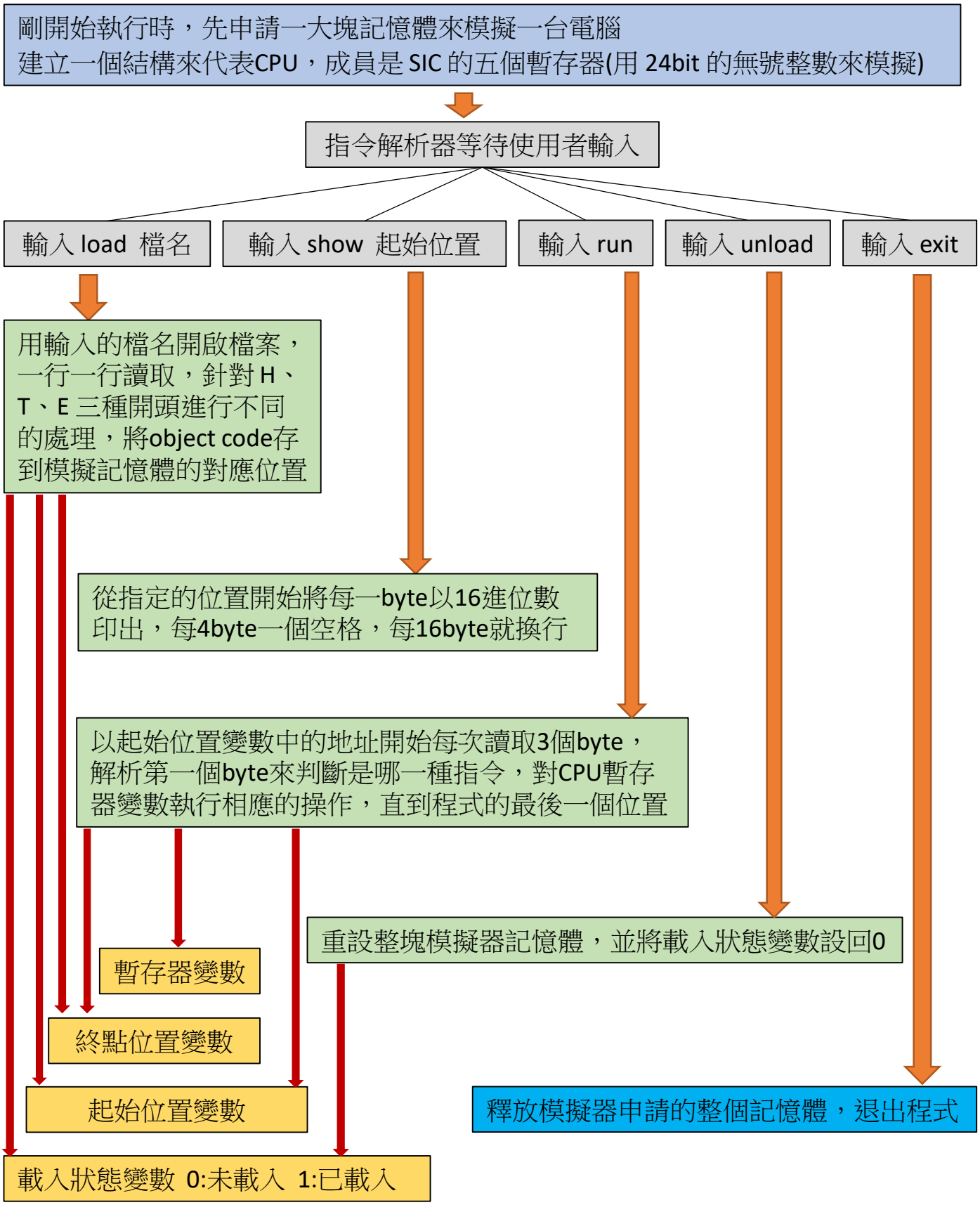
系統程式期末報告
SIC simulator

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



```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include"tool.h"


typedef unsigned char uint8_t;


typedef struct cpu_t{
    unsigned int A : 24;
    unsigned int X : 24;
    unsigned int L : 24;
    unsigned int PC : 24;
    unsigned int SW : 24;
}cpu_t;

void show(void * memory, char * begin, char * end);

void load(char * target, void * memory, int * program_begin,  int * program_end, int
* loaded);

void run(int * program_begin, int * program_end, void * memory, cpu_t * cpu);
```

```h
#ifndef __TOOL_H_
#define __TOOL_H_

#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int exponent_Int(const int base, int n);

int hex_to_dex(char *hex);

int hex_to_dex_c(char hex);

#endif
```

```c
#include"simulator.h"



int main(int argc, char *argv[]){
    char input[100];
    char temp1[100];
    char temp2[100];
    void * memory = NULL;
    int system_size = 0;
    int loaded = 0;
    int program_begin = 0;
    int program_end = 0;
    if(argv[1] == 0)
    {
        memory = malloc(hex_to_dex("5000"));
        memset(memory, '.', hex_to_dex("5000"));
        system_size = hex_to_dex("5000");
        printf("system memory from 0x0 ~ 0x%X\n", system_size);
    }
    else
    {
        system_size = hex_to_dex(argv[1]);
        printf("system memory from 0x0 ~ 0x%X\n", system_size);
        memory = malloc(system_size);
        memset(memory, '.',  system_size);

    }

    temp2[0] = '0';
    temp2[1] = '\0';
    temp1[0] = '\0';

    //CPU
    cpu_t cpu;
    cpu.A = 0;
    cpu.X = 0;
    cpu.L = 0;
    cpu.PC = 0;
    cpu.SW = 0;




    while(1)
    {
        printf(">>> ");
        fgets(input, 100, stdin);
        sscanf(input, "%s %s", temp1, temp2);

        if(strcmp(temp1, "exit") == 0)
        {
            free(memory);
            break;
        }

        else if(strcmp(temp1, "show") == 0)
        {
            show(memory, temp2, argv[1]);
            temp2[0] = '0';
            temp2[1] = '\0';
            temp1[0] = '\0';
        }

        else if(strcmp(temp1, "load") == 0)
        {
            if(loaded == 0)
            {
                load(temp2, memory, &program_begin, &program_end, &loaded);
            }
            else
            {
                printf("error , an object code has been load\n");
            }


            temp2[0] = '0';
            temp2[1] = '\0';
```

```
            temp1[0] = '\0';
        }

        else if(strcmp(temp1, "unload") == 0)
        {
            memset(memory, '.',  system_size);
            cpu.A = 0;
            cpu.X = 0;
            cpu.L = 0;
            cpu.PC = 0;
            cpu.SW = 0;
            loaded = 0;
            temp2[0] = '0';
            temp2[1] = '\0';
            temp1[0] = '\0';
            printf("A : %06X  X : %06X  L : %06X  PC : %06X  SW : %06X\n\n", cpu.A,
cpu.X, cpu.L, cpu.PC, cpu.SW);
        }

        else if(strcmp(temp1, "run") == 0)
        {
            run(&program_begin, &program_end, memory, &cpu);
            temp2[0] = '0';
            temp2[1] = '\0';
            temp1[0] = '\0';
            printf("A : %06X  X : %06X  L : %06X  PC : %06X  SW : %06X\n\n", cpu.A,
cpu.X, cpu.L, cpu.PC, cpu.SW);
        }

        else
        {
            printf("command not find\n");
        }
    }
}
```

```c
#include"simulator.h"

void load(char * target, void * memory, int * program_begin,  int * program_end, int
* loaded)
{
    char buffer[100];
    FILE * load_obj = fopen(target,"r");
    if(load_obj == NULL)
    {
        printf("error\n");
    }
    else
    {
        *loaded = 1;
        char begin[20];
        int size = 0;
        while(fgets(buffer, 100, load_obj))
        {
            if(*buffer == 'H')
            {
                for(int i = 0 ; i < 6 ; i++)
                {
                    begin[i] = buffer[i + 7];
                }
                begin[6] = '\0';
                printf("%d ", hex_to_dex(begin));
                *program_begin = hex_to_dex(begin);

                size = hex_to_dex(buffer+13);
                printf("%d \n", size);
                *program_end = *program_begin + size;

            }

            if(*buffer == 'T')
            {
                for(int i = 0 ; i < 6 ; i++)
                {
                    begin[i] = buffer[i + 1];
                }
                begin[6] = '\0';

                printf("%X ", hex_to_dex(begin));
                int memory_count = hex_to_dex(begin);
                int state = 0;

                for(int i = 9 ; i < strlen(buffer) - 1 ; i++)
                {
                    if(state == 0)
                    {
                        *((uint8_t *)memory + memory_count) =
(hex_to_dex_c(buffer[i]) << 4) ;
                        printf("%c", buffer[i]);
                        state = 1;
                    }
                    else if(state == 1)
                    {
                        *((uint8_t *)memory + memory_count) |=
(hex_to_dex_c(buffer[i]) << 0) ;
                        memory_count++;
                        printf("%c", buffer[i]);
                        state = 0;
                    }
                }
                printf("\n");
            }
        }
        fclose(load_obj);
    }
}
```

```c
#include"simulator.h"


void show(void * memory, char * begin, char * end)
{

    int offset = 0;
    if(hex_to_dex(begin) <= hex_to_dex(end)-1)
    {
        if(hex_to_dex(begin) % 16 != 0)
        {
            printf("%04X   ", hex_to_dex(begin));
            for(int i = 0 ; i < (hex_to_dex(begin) % 16) ; i++)
            {
                printf("  ");
                offset++;
                if(offset == 4 || offset == 8 || offset == 12)
                {
                    printf("   ");
                }
            }
        }

        for(int i = hex_to_dex(begin); i < hex_to_dex(end); i++)
        {
            if(i % 16 == 0)
            {
                offset = 0;
                printf("%04X   ", i);
            }

            if(*((char *)(memory+i)) == '.')
            {
                printf("..");
            }
            else
            {
                printf("%02X", *((uint8_t *)(memory+i)));
            }

            offset++;

            if(offset == 4 || offset == 8 || offset == 12)
            {
                printf("   ");
            }

            if(offset == 16)
            {
                printf("\n");
            }
        }
        offset = 0;
        printf("\n");
    }
    else
    {
        printf("error %d\n", hex_to_dex(begin));
    }
}
```

```c
#include"simulator.h"



void run(int * program_begin, int * program_end, void * memory, cpu_t * cpu)
{
    printf("%X %X\n", *program_begin, *program_end);
    cpu->PC = *program_begin;
    int temp = 0;
    int temp2 = 0;
    int temp3 = 0;
    int x_mod = 0;
    FILE * input = NULL;
    FILE * output = NULL;
    input = fopen("F1.txt", "r");
    output = fopen("F5.txt", "w");
    cpu->L = *program_end;


    while(cpu->PC < *program_end)
    {
        x_mod = 0;
        temp = 0;
        temp2 = 0;
        temp3 = 0;
        if(*((uint8_t *)memory + cpu->PC + 1) >= 128)
        {
            x_mod = 1;
        }

        switch(*((uint8_t *)memory + cpu->PC))
        {
            case 24:
                printf("ADD     ");
                temp = *((uint8_t *)memory + cpu->PC + 1);
                temp = temp << 8;
                temp |= *((uint8_t *)memory + cpu->PC + 2);

                temp2 = *((uint8_t *)memory + temp + 2);
                cpu->A += temp2;

                temp2 = *((uint8_t *)memory + temp + 1);
                temp2 = temp2 << 8;
                cpu->A += temp2;

                temp2 = *((uint8_t *)memory + temp);
                temp2 = temp2 << 16;
                cpu->A += temp2;


                cpu->PC += 3;
                break;

            case 52:
                printf("JGT     ");
                cpu->PC += 3;
                break;

            case 0:
                printf("LDA     ");
                temp = *((uint8_t *)memory + cpu->PC + 1);
                temp = temp << 8;
                temp |= *((uint8_t *)memory + cpu->PC + 2);

                if(*((uint8_t *)memory + temp) == '.')
                {
                    cpu->A = 0;
                }
                else
                {
                    cpu->A = *((uint8_t *)memory + temp);
                }

                cpu->A = cpu->A << 8;

                if(*((uint8_t *)memory + temp + 1) == '.')
                {
                    cpu->A |= 0;
```

```
            }
            else
            {
                cpu->A |= *((uint8_t *)memory + temp + 1);
            }

            cpu->A = cpu->A << 8;

            if(*((uint8_t *)memory + temp + 2) == '.')
            {
                cpu->A |= 0;
            }
            else
            {
                cpu->A |= *((uint8_t *)memory + temp + 2);
            }



            cpu->PC += 3;
            break;

        case 4:
            printf("LDX     ");
            temp = *((uint8_t *)memory + cpu->PC + 1);
            temp = temp << 8;
            temp |= *((uint8_t *)memory + cpu->PC + 2);

            if(*((uint8_t *)memory + temp) == '.')
            {
                cpu->X = 0;
            }
            else
            {
                cpu->X = *((uint8_t *)memory + temp);
            }

            cpu->X = cpu->X << 8;

            if(*((uint8_t *)memory + temp + 1) == '.')
            {
                cpu->X |= 0;
            }
            else
            {
                cpu->X |= *((uint8_t *)memory + temp + 1);
            }

            cpu->X = cpu->X << 8;

            if(*((uint8_t *)memory + temp + 2) == '.')
            {
                cpu->X |= 0;
            }
            else
            {
                cpu->X |= *((uint8_t *)memory + temp + 2);
            }


            cpu->PC += 3;
            break;

        case 216:
            printf("RD      ");
            temp = *((uint8_t *)memory + cpu->PC + 1);
            temp = temp << 8;
            temp |= *((uint8_t *)memory + cpu->PC + 2);


            //cpu->A = *((uint8_t *)memory + temp);

            temp2 = getc(input);
            if(temp2 == -1)
            {
                cpu->A = 0;
            }
            else
            {
```

```
                cpu->A = (unsigned int)temp2;
            }



            cpu->PC += 3;
            break;

        case 220:
            printf("WD      ");
            temp = *((uint8_t *)memory + cpu->PC + 1);
            temp = temp << 8;
            temp |= *((uint8_t *)memory + cpu->PC + 2);

            //*((uint8_t *)memory + temp) = cpu->A;
            putc(cpu->A, output);

            cpu->PC += 3;
            break;

        case 84:
            printf("STCH    ");
            temp = *((uint8_t *)memory + cpu->PC + 1);
            temp = temp << 8;
            temp |= *((uint8_t *)memory + cpu->PC + 2);

            if(x_mod == 1)
            {
                temp = temp ^ (1 << 15);
                *((uint8_t *)memory + temp + cpu->X) = cpu->A;
            }
            else
            {
                *((uint8_t *)memory + temp) = cpu->A;
            }

            cpu->PC += 3;
            break;

        case 16:
            printf("STX     ");
            temp = *((uint8_t *)memory + cpu->PC + 1);
            temp = temp  << 8;
            temp |= *((uint8_t *)memory + cpu->PC + 2);

            temp2 = cpu->X;

            *((uint8_t *)memory + temp + 2) = temp2;
            temp2 = temp2 >> 8;
            *((uint8_t *)memory + temp + 1) = temp2;
            temp2 = temp2 >> 8;
            *((uint8_t *)memory + temp) = temp2;


            cpu->PC += 3;

            break;

        case 12:
            printf("STA     ");
            temp = *((uint8_t *)memory + cpu->PC + 1);
            temp = temp  << 8;
            temp |= *((uint8_t *)memory + cpu->PC + 2);

            temp2 = cpu->A;

            *((uint8_t *)memory + temp + 2) = temp2;
            temp2 = temp2 >> 8;
            *((uint8_t *)memory + temp + 1) = temp2;
            temp2 = temp2 >> 8;
            *((uint8_t *)memory + temp) = temp2;


            cpu->PC += 3;

            break;

        case 44:
            printf("TIX     ");
```

```
                cpu->X++;

                temp = *((uint8_t *)memory + cpu->PC + 1);
                temp = temp << 8;
                temp |= *((uint8_t *)memory + cpu->PC + 2);

                temp2 = *((uint8_t *)memory + temp);
                temp2 = temp2 << 8;
                temp2 |= *((uint8_t *)memory + temp + 1);
                temp2 = temp2 << 8;
                temp2 |= *((uint8_t *)memory + temp + 2);


                if(cpu->X == temp2)
                {
                    cpu->SW = '=';
                }
                else if(cpu->X < temp2)
                {
                    cpu->SW = '<';
                }
                else
                {
                    cpu->SW = '>';
                }

                cpu->PC += 3;
                break;

            case 64:
                printf("AND     ");
                cpu->PC += 3;
                break;

            case 60:
                printf("J       ");
                temp = cpu->PC;
                cpu->PC = *((uint8_t *)memory + temp + 1);
                cpu->PC = cpu->PC  << 8;
                cpu->PC |= *((uint8_t *)memory + temp + 2);
                break;

            case 56:
                printf("JLT     ");
                temp = cpu->PC;
                if(cpu->SW == '<')
                {
                    cpu->PC = *((uint8_t *)memory + temp + 1);
                    cpu->PC = cpu->PC  << 8;
                    cpu->PC |= *((uint8_t *)memory + temp + 2);

                }
                else
                {
                    cpu->PC += 3;
                }
                break;

            case 80:
                printf("LDCH    ");
                temp = *((uint8_t *)memory + cpu->PC + 1);
                temp = temp << 8;
                temp |= *((uint8_t *)memory + cpu->PC + 2);

                if(x_mod == 1)
                {
                    temp = temp ^ (1 << 15);
                    cpu->A = *((uint8_t *)memory + temp + cpu->X);
                }
                else
                {
                    cpu->A = *((uint8_t *)memory + temp);
                }

                cpu->PC += 3;
                break;

            case 32:
                printf("MUL     ");
```

```
            temp = *((uint8_t *)memory + cpu->PC + 1);
            temp = temp << 8;
            temp |= *((uint8_t *)memory + cpu->PC + 2);

            temp3 = cpu->A;
            cpu->A = 0;

            temp2 = *((uint8_t *)memory + temp + 2);
            cpu->A += temp3 * temp2;

            temp2 = *((uint8_t *)memory + temp + 1);
            temp2 = temp2 << 8;
            cpu->A += temp3 * temp2;

            temp2 = *((uint8_t *)memory + temp);
            temp2 = temp2 << 16;
            cpu->A += temp3 * temp2;


            cpu->PC += 3;
            break;

        case 76:
            printf("RSUB    ");
            cpu->PC = cpu->L;
            break;

        case 20:
            printf("STL     ");
            temp = *((uint8_t *)memory + cpu->PC + 1);
            temp = temp  << 8;
            temp |= *((uint8_t *)memory + cpu->PC + 2);

            temp2 = cpu->L;

            *((uint8_t *)memory + temp + 2) = temp2;
            temp2 = temp2 >> 8;
            *((uint8_t *)memory + temp + 1) = temp2;
            temp2 = temp2 >> 8;
            *((uint8_t *)memory + temp) = temp2;



            cpu->PC += 3;
            break;

        case 28:
            printf("SUB     ");
            temp = *((uint8_t *)memory + cpu->PC + 1);
            temp = temp << 8;
            temp |= *((uint8_t *)memory + cpu->PC + 2);

            temp2 = *((uint8_t *)memory + temp);
            temp2 = temp2 << 8;

            temp2 = *((uint8_t *)memory + temp + 1);
            temp2 = temp2 << 8;

            temp2 = *((uint8_t *)memory + temp + 2);

            cpu->A = cpu->A - temp2;


            cpu->PC += 3;
            break;



        case 40:
            printf("COMP    ");
            temp = *((uint8_t *)memory + cpu->PC + 1);
            temp = temp << 8;
            temp |= *((uint8_t *)memory + cpu->PC + 2);

            temp2 = *((uint8_t *)memory + temp);
            temp2 = temp2 << 8;
            temp2 |= *((uint8_t *)memory + temp + 1);
            temp2 = temp2 << 8;
            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
```

```
                cpu->PC += 3;
                break;

            case 68:
                printf("OR      ");
                cpu->PC += 3;
                break;

            case 232:
                printf("STSW    ");
                cpu->PC += 3;
                break;

            case 224:
                printf("TD      ");
                temp = *((uint8_t *)memory + cpu->PC + 1);
                temp = temp << 8;
                temp |= *((uint8_t *)memory + cpu->PC + 2);


            temp2 = *((uint8_t *)memory + temp);
                //printf("%02X\n", temp2);

                if(temp2 == hex_to_dex("F1"))
                {
                    if(input != NULL)
                    {
                        cpu->SW = '<';
                    }
                    else
                    {
                        cpu->SW = '=';
                    }
                }

                if(temp2 == hex_to_dex("05"))
                {
                    if(output != NULL)
                    {
                        cpu->SW = '<';
                    }
                    else
                    {
                        cpu->SW = '=';
                    }
                }

                /*
                if(*((uint8_t *)memory + temp) == '.')
                {
                    cpu->SW = '=';
                }
                else
                {
                    cpu->SW = '<';
                }
                */

                cpu->PC += 3;
                break;

            default:
                break;
        }
        printf("A : %06X  X : %06X  L : %06X  PC : %06X  SW : %06X\n\n", cpu->A, cpu-
>X, cpu->L, cpu->PC, cpu->SW);
    }
    fclose(input);
    fclose(output);

    printf("\n");
}
```

```c
#include"tool.h"


//將指數乘開的函數
int exponent_Int(const int base, int n)
{
    int p = base;
    if(n == 0)
    {
        p = 1;
    }
    else
    {
        for (int i = 1; i < n; i++)
        {
            p *= base;
        }
    }
    return p;
}


//將讀取的字串視為16進位，並轉換為10進位數字的函數
int hex_to_dex(char *hex)
{
    char * char_temp = (char *)malloc(strlen(hex));
    char * temp_begin = char_temp;
    strcpy(char_temp, hex);
    char_temp = strtok(char_temp, "\n");
    char temp[2];
    int total = 0;
    int count = strlen(char_temp);

    while(count-- && count >= 0)
    {
        sprintf(temp, "%c", *char_temp);
        if(strcmp(temp, "A") == 0){total += exponent_Int(16, count)*10;}
        if(strcmp(temp, "B") == 0){total += exponent_Int(16, count)*11;}
        if(strcmp(temp, "C") == 0){total += exponent_Int(16, count)*12;}
        if(strcmp(temp, "D") == 0){total += exponent_Int(16, count)*13;}
        if(strcmp(temp, "E") == 0){total += exponent_Int(16, count)*14;}
        if(strcmp(temp, "F") == 0){total += exponent_Int(16, count)*15;}
        else{total += exponent_Int(16, count)*atoi(temp);}
        char_temp++;
    }
    free(temp_begin);
    return total;
}

//將讀取的字元視為16進位，並轉換為10進位數字的函數
int hex_to_dex_c(char hex)
{
    if(hex == 'A'){return 10;}
    if(hex == 'B'){return 11;}
    if(hex == 'C'){return 12;}
    if(hex == 'D'){return 13;}
    if(hex == 'E'){return 14;}
    if(hex == 'F'){return 15;}
    if(hex == '9'){return 9;}
    if(hex == '8'){return 8;}
    if(hex == '7'){return 7;}
    if(hex == '6'){return 6;}
    if(hex == '5'){return 5;}
    if(hex == '4'){return 4;}
    if(hex == '3'){return 3;}
    if(hex == '2'){return 2;}
    if(hex == '1'){return 1;}
    if(hex == '0'){return 0;}
    return 0;
}
```

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 14103348203900103628103030101548206 13C100300102A0C103900102D
101E 0C10364820610810334C0000454F46000003000000
2039 041030001030E0205D30203FD8205D28103030205754903 92C205E38203F
2057 1010364C0000F1001000041030E020793020645090 39DC20792C1036
2073 3820644C000005
>>>
```

用show 顯示記憶體中的內容

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

```
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   ........   ........   ........   ........
```

```
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。