Report 6

Algorithm Explanation

I set up two arrays to store data. One is to store the number of code blocks (there are spaces before and after one code block), and the other is to store the starting address of each instruction in the machine code in the row of the sink code.

```
char preblock[30005][55];
int beginline[10005];
```

I use map to store label and its corresponding address.

```
map<string,int>label;
```

This algorithm contains 3 parts: input, first execution and second execution.

- input part:

  Read code blocks separated by spaces or swap lines into the preblock array. If the .STRINGZ instruction is read, manually store the next thing in double quotation marks into the array.

- first execution:
  - Output the second code block, which is the starting address of the program.
  - Divide the instructions into three, two, one suffix, no suffix,.Bklw instruction and There are six types of stringz instructions, each of which is processed differently, and their starting addresses are stored in the beginline array.
  - If the instruction is .BKLW, beginline array subscript plus number after .BKLW. Else if it is .STRINGZ, let the next length of .STRINGZ the beginline array subscript be -1.
- second execution:

instruction name	function it used
ADD	printregister/printsharp/printx
AND	printregister/printsharp/printx
NOT	printregister
LD	printregister/printsharp/printoff
LDR	printregister/printsharp/printx
LDI	printregister/printsharp/printoff
LEA	printregister/printsharp/printoff
ST	printregister/printsharp/printoff
STR	printregister/printsharp/printx
STI	printregister/printsharp/printoff
TRAP	hextodec
BR	printsharp
JMP	printregister
JSR	printsharp/printoff/printregister
RTI	/
.FILL	printsharp/printx
.BKLW	/
.STRINGZ	dectobin

Evey function and its effect

name	input	output	function
hextodec	char *s	int dec	Hexadecimal to decimal
printregister	int r	/	Output register three bit binary number
_dectobin	int dec,int bit	/	Decimal to binary output (complement)
dectobin	int dec,int bit	/	Decimal to binary output
printsharp	int loc,int bit	/	output string begin with #
printoff	int pc,int bit	/	output offset9
printx	int loc,int bit	/	output string begin with x

Essential parts

hextodec

```
int hextodec(char* s) {
    int i=0,dec=0;
    if(s[0]=='x'||s[0]=='X') i=1;
    while(s[i]!=0) {
        int tmp=0;
        if(s[i]>='0'&&s[i]<='9')tmp=s[i]-'0';
        if(s[i]>='A'&&s[i]<='F')tmp=s[i]-'A'+10;
        if(s[i]>='a'&&s[i]<='f')tmp=s[i]-'a'+10;
        dec=dec*16+tmp;
        i++;
    }
    return dec;
}
```

_dectobin

```
void _dectobin(int dec,int bit) {
    int print[20];
    memset(print,0,sizeof(print));
    int i;
    for(i=0; i<bit; i++) {
        print[bit-1-i]=1-dec%2;
        dec/=2;
    }
    print[bit-1]++;
    for(i=0; i<bit-1; i++) {
        if(print[bit-1-i]==2) {
            print[bit-1-i]=0;
            print[bit-i-2]++;
        }
    }
    if(print[0]==2)print[0]=0;
    for(i=0; i<bit; i++) {
        printf("%d",print[i]);
    }
}
```

printsharp

```
void printsharp(int loc,int bit) {
//  printf("\n%s\n",preblock[loc]);
    if(preblock[loc][1]=='-') {
        int length=strlen(preblock[loc]);
        length-=2;
        int k=0,sum=0;
        for(k=0; k<length; k++) {
            sum*=10;
            sum+=preblock[loc][2+k]-'0';
        }
        _dectobin(sum,bit);
    } else {
        int length=strlen(preblock[loc]);
        length-=1;
        int k=0,sum=0;
        for(k=0; k<length; k++) {
            sum*=10;
            sum+=preblock[loc][1+k]-'0';
        }
//      printf("\n%d\n",sum);
        dectobin(sum,bit);
    }
}
```

out put offset

```
if(preblock[beginline[j]+2][0]=='#'){
    printsharp(beginline[j]+2,9);
    printf("\n");
}else{
    int pc=label[preblock[beginline[j]+2]];
    pc-=j+1;
    printoff(pc,9);
    printf("\n");
}
```

ADD instruction

```
if(!strcmp(preblock[beginline[j]],"ADD")) {
    printf("0001");
    printregister(preblock[beginline[j]+1][1]-'0');
    printregister(preblock[beginline[j]+2][1]-'0');
    if(preblock[beginline[j]+3][0]=='#') {
    printf("1");
    printsharp(beginline[j]+3,5);
} else if(preblock[beginline[j]+3][0]=='R'){
    printf("000");
    printregister(preblock[beginline[j]+3][1]-'0');
}else{
    printf("1");
    printx(beginline[j]+3,5);
}
    printf("\n");
}
```

.STRINGZ instruction

```c
if(!strcmp(preblock[beginline[j]],".STRINGZ")){
    int length=strlen(preblock[beginline[j]+1]);
    int k;
    for(k=0;k<length;k++){
        int x=preblock[beginline[j]+1][k];
        dectobin(x,16);
        printf("\n");
    }
    printf("0000000000000000\n");
}
```

Q&A

NULL