```
1
2
3   // ******************** INSTRUCTIONS ********************
4
5       void load() {
6          printf("\tLoad\n");
7          if (mode == 1) {
8             Registers[reg] = address;
9          }
10         else {
11            Registers[reg] = (short)mainMemory[address];
12         }
13
14         changeCondition(reg);
15      }
16
17      void store() {
18         printf("\tStore\n");
19         mainMemory[address] = (unsigned short)Registers[reg];
20
21         changeCondition(reg);
22      }
23
24      void add() {
25         printf("\tAdd\n");
26         if (mode == 1) {
27            Registers[reg] = Registers[0] + address;
28         }
29         else {
30            Registers[reg] = Registers[0] + (short)mainMemory[address];
31         }
32
33         changeCondition(reg);
34      }
35
36      void sub() {
37         printf("\tSubtract\n");
38         if (mode == 1) {
39            Registers[reg] = Registers[0] - address;
40         }
41         else {
42            Registers[reg] = Registers[0] - (short)mainMemory[address];
43         }
44
45         changeCondition(reg);
46      }
47
48      void adr() {
49         printf("\tAdd Register\n");
50         Registers[0] = Registers[0] + Registers[reg];
51
52         changeCondition(0);
53      }
```

```
54
55      void sur() {
56         printf("\tSubtract Register\n");
57         Registers[0] = Registers[0] - Registers[reg];
58
59         changeCondition(0);
60      }
61
62      void and() {
63         printf("\tAnd\n");
64         if (mode == 1) {
65            Registers[reg] = Registers[0] & address;
66         }
67         else {
68            Registers[reg] = Registers[0] & (short)mainMemory[address];
69         }
70
71         changeCondition(reg);
72      }
73
74      void or() {
75         printf("\tOr\n");
76         if (mode == 1) {
77            Registers[reg] = Registers[0] | address;
78         }
79         else {
80            Registers[reg] = Registers[0] | (short)mainMemory[address];
81         }
82
83         changeCondition(reg);
84      }
85
86      void not() {
87         printf("\tNot\n");
88
89         Registers[reg] = ~Registers[reg];
90
91         changeCondition(reg);
92      }
93
94      void jmp() {
95         printf("\tJump\n");
96         PC = (unsigned short)address;
97      }
98
99      void jeq() {
100        printf("\tJump Equal\n");
101        if (CC == 2) PC = (unsigned short)address;
102     }
103
104     void jgt() {
105        printf("\tJump Greater\n");
106        if (CC == 1) PC = (unsigned short)address;
```

```
107      }
108
109      void jlt() {
110         printf("\tJump Less\n");
111         if (CC == 4) PC = (unsigned short)address;
112      }
113
114      void compare() {
115         printf("\tCompare\n");
116         if (Registers[reg] > 0) {
117            CC = 1;
118         }
119         else if(Registers[reg] == 0) {
120            CC = 2;
121         }
122         else if(Registers[reg] < 0) {
123            CC = 4;
124         }
125         else {}
126      }
127
128      void clear() {
129         printf("\tClear\n");
130         Registers[reg] = 0;
131
132         changeCondition(reg);
133      }
134
135      void halt() {
136         haltFlag = true;
137         printf("\tHalt\n");
138
139         printf("Execution complete.\n");
140      }
141
```