

浙江大学

本科实验报告

课程名称:	计算机组成
姓名:	臧可
学院:	计算机科学与技术学院
系:	计算机科学与技术系
专业:	计算机科学与技术
学号:	3180102095
指导教师:	姜晓红

2020年 4月 29日

一、实验目的和要求

- Supporting the following 18 MIPS instructions: Add, Sub, And, Or, Addi, Ori, Sll, Srl, Lw, Sw, Lui, Slt, Slti, Beq, Bne, J, Jal, Jr.
- input a MIPS assembly program, output its MIPS machine code in hexadecimal code.
- Input a MIPS machine code in hexadecimal code, output its MIPS assembly program with PC value.

二、实验内容和原理

1. class

mips: show assembler operation

```
1  class mips
2  {
3  public:
4      mips(string file1, string file2);
5  private:
6      struct Jump { //Record jump instructions
7          int locate; //Number of lines where the instruction is located
8          string type; //Instruction name
9          string Label; //Label name
10     } jump[100];
11     int line; //File lines
12     int structsize; //Number of jump instructions
13     ofstream outfile;
```

```

14
15     void opToHex(string op); //mips command to hexadecimal
16     string toLower(string s); //Uppercase to lowercase
17     int* getReg(char* p); //Uppercase to lowercase
18     int* iToBin(int reg, int width); //int to binary
19     int* iToBinOff(int reg, int width); //Calculate the offset
20     int* numToBin(char* p); //address behind J_type to binary
21     char* binToHex(int* b); //32-bit binary to hexadecimal
22 };

```

Invert: show disassembler operation

```

1  class Invert
2  {
3  public:
4      Invert(string file); //Read instruction file
5      void Convert(string file1, string file2);
6  private:
7      struct ins { //Store instruction
8          string type; //Instruction type
9          string op; //Instruction opcode
10         string name; //Instruction name
11     };
12     vector<ins> R_type, I_type, J_type; //Store 3 kinds of instructions
13     string findName(const vector<ins>& v, string s); //Find the
14     string corresponding command name
15     int binToInt(string s); //Binary to int
16     static bool cmp(ins a, ins b); //Used for instruction queue sorting
17     string hexToBin(string h); //4-bit hexadecimal to 16-bit binary
18     string iToSymbol(int t); //int to register symbol
19     char* iToHex(int reg); //int to 8-bit hexadecimal
20     char* binToHex(int* b); //32-bit binary to 8-bit hexadecimal
21 };

```

lineProcess: Realize the removal of spaces and comments on files

```

1  class lineProcess
2  {
3  private:
4      string str; //the input line
5      string comment_str; //Comment string
6  public:
7      lineProcess(string& s, string c);
8      string Cut(); //Cut unnecessary parts
9  };

```

2. Main function

```

1  #include "lineProcess.h"

```

```

2  #include "mips.h"
3  #include "Invert.h"
4
5  void Assembler();
6  void Dissembler();
7
8  int main() {
9      int AsOrDiss;
10     cout << "Choose assembler or dissembler, 1 is assembler and 0 is
dissembler" << endl;
11     cin >> AsOrDiss;
12     if (AsOrDiss) Assembler();
13     else Dissembler();
14
15     return 0;
16 }
17
18 void Assembler() {
19     string filename;
20     cout << "Please input the name of the file you want to open." << endl;
21     cin >> filename;
22     mips a(filename, "mips.coe");
23     cout << "You can find the answer in the file \"mips.coe\"." << endl;
24 }
25
26 void Dissembler() {
27     string filename;
28     cout << "Please input the name of the file you want to open." << endl;
29     cin >> filename;
30     Invert d("instruction.txt");
31     d.Convert(filename, "invert.txt");
32     cout << "You can find the answer in the file \"invert.txt\"." << endl;
33 }

```

3. .cpp file

mips.cpp

```

1  #include "mips.h"
2  #include "lineProcess.h"
3
4  mips::mips(string file1, string file2) {
5      ifstream infile(file1);
6      outfile.open(file2);
7      int i, j;
8      string str;
9
10     if (!infile) {
11         cout << "Open error!" << endl;
12         exit(1);
13     }
14     outfile << "memory_initialization_radix=16;" << endl;
15     outfile << "memory_initialization_vector = " << endl;
16     cnt = 0;
17

```

```

18     i = -1;
19     line = -1;
20
21     while (getline(infile, str))//Traverse the file for the first time,
record the jump instruction and label information
22     {
23         lineProcess linep(str, "//");//Remove multiple spaces and tabs at
the beginning and end of lines
24
25         str = linep.Cut();
26         if (str.empty()) continue;//continue if line is empty
27         line++;
28         istringstream iss(str);
29         string op1, op2, op3, op4;
30         string::size_type position;
31         iss >> op1;
32         position = op1.find(":");
33         if (op1 == "j" || op1 == "jal") {
34             iss >> op2;
35             i++;
36             jump[i].locate = line;
37             jump[i].type = op1;
38             jump[i].Label = op2;
39         }
40         else if (op1 == "beq" || op1 == "bne") {
41             iss >> op2 >> op3 >> op4;
42             i++;
43             jump[i].locate = line;
44             jump[i].type = op1;
45             jump[i].Label = op4;
46         }
47         else if (position != op1.npos) {
48             op1.erase(position);
49             i++;
50             jump[i].locate = line;
51             jump[i].type = "tab";
52             jump[i].Label = op1;
53             line--;
54         }
55     }
56
57     structsize = i;
58
59     infile.clear();
60     infile.seekg(0, ios::beg);
61     i = 0;
62     line = -1;
63     while (getline(infile, str)) {
64         lineProcess linep(str, "//");//Remove multiple spaces and tabs at
the beginning and end of lines
65         str = linep.Cut();
66         if (str.empty()) continue;//continue if line is empty
67         istringstream iss(str);
68         string op;
69         iss >> op;
70         if (op.back() == ':') continue;//Skip when reading label
71         line++;
72         opToHex(str);

```

```

73     }
74
75     infile.close();
76 }
77
78 void mips::opToHex(string op) { //mips command to hexadecimal
79     int i, j;
80     string p0;
81     string p1;
82     char* p2 = new char[10];
83     int* nowbin = new int[32]; //Record the 32-bit binary machine code
corresponding to the mips instruction
84     char* nowhex = new char[8]; //Record the 8-digit hexadecimal machine
code corresponding to the mips instruction
85     istringstream iss(op);
86     op = toLower(op);
87     iss >> p0;
88     for (i = 0; i < 32; ++i)
89         nowbin[i] = 0;
90     if (p0 == "add" || p0 == "sub" || p0 == "and" || p0 == "or" || p0 ==
"slt" || p0 == "nor") {
91         iss >> p1;
92         strcpy(p2, p1.c_str());
93         int* rd = getReg(p2);
94         iss >> p1;
95         strcpy(p2, p1.c_str());
96         int* rs = getReg(p2);
97         iss >> p1;
98         strcpy(p2, p1.c_str());
99         int* rt = getReg(p2);
100        for (i = 6; i < 11; ++i) {
101            nowbin[i] = rs[i - 6];
102            nowbin[i + 5] = rt[i - 6];
103            nowbin[i + 10] = rd[i - 6];
104        }
105        nowbin[26] = 1;
106        if (p0 == "sub")
107            nowbin[30] = 1;
108        else if (p0 == "and")
109            nowbin[29] = 1;
110        else if (p0 == "or")
111            nowbin[29] = nowbin[31] = 1;
112        else if (p0 == "slt")
113            nowbin[28] = nowbin[30] = 1;
114        else if (p0 == "nor")
115            nowbin[29] = nowbin[30] = nowbin[31] = 1;
116
117        nowhex = binToHex(nowbin);
118
119        for (i = 0; i < 8; ++i) {
120            outfile << nowhex[i];
121        }
122        outfile << ", ";
123        cnt++;
124    }
125    else if (p0 == "sll" || p0 == "srl") {
126        iss >> p1;
127        strcpy(p2, p1.c_str());

```

```

128     int* rd = getReg(p2);
129     iss >> p1;
130     strcpy(p2, p1.c_str());
131     int* rt = getReg(p2);
132     iss >> p1;
133     strcpy(p2, p1.c_str());
134     int t = atoi(p2);
135     int* shamt = iToBin(t, 5);
136     for (i = 11; i < 16; ++i) {
137         nowbin[i] = rt[i - 11];
138         nowbin[i + 5] = rd[i - 11];
139         nowbin[i + 10] = shamt[i - 11];
140     }
141     if (p0 == "srl")
142         nowbin[30] = 1;
143
144     nowhex = binToHex(nowbin);
145
146     for (i = 0; i < 8; ++i) {
147         outfile << nowhex[i];
148     }
149     outfile << ", ";
150     cnt++;
151 }
152 else if (p0 == "addi" || p0 == "ori" || p0 == "slti") {
153     iss >> p1;
154     strcpy(p2, p1.c_str());
155     int* rt = getReg(p2);
156     iss >> p1;
157     strcpy(p2, p1.c_str());
158     int* rs = getReg(p2);
159     iss >> p1;
160     strcpy(p2, p1.c_str());
161     int* imm = numToBin(p2);
162     if (p0 == "addi")
163         nowbin[2] = 1;
164     else if (p0 == "ori")
165         nowbin[0] = nowbin[2] = nowbin[4] = nowbin[5] = 1;
166     else if (p0 == "slti")
167     {
168         nowbin[2] = nowbin[4] = 1;
169     }
170     for (i = 6; i < 11; ++i) {
171         nowbin[i] = rs[i - 6];
172         nowbin[i + 5] = rt[i - 6];
173     }
174     for (i = 16; i < 32; ++i)
175         nowbin[i] = imm[i - 16];
176
177     nowhex = binToHex(nowbin);
178     for (i = 0; i < 8; ++i) {
179         outfile << nowhex[i];
180     }
181     outfile << ", ";
182     cnt++;
183 }
184
185 else if (p0 == "beq" || p0 == "bne") {

```

```

186     iss >> p1;
187     strcpy(p2, p1.c_str());
188     int* rs = getReg(p2);
189     iss >> p1;
190     strcpy(p2, p1.c_str());
191     int* rt = getReg(p2);
192     iss >> p1;
193     int offset;
194     for (j = 0; j <= structsize; ++j) {
195         if (jump[j].locate == line && jump[j].type == p0)
196             break;
197     }
198     for (i = 0; i <= structsize; ++i) {
199         if (jump[i].Label == jump[j].Label && jump[i].type == "tab") {
200             offset = jump[i].locate - line - 1;
201             offset *= 4;
202             break;
203         }
204     }
205     int* imm = iToBinOff(offset, 16); //calculate the offset
206
207     nowbin[3] = 1;
208     if (p0 == "bne")
209         nowbin[5] = 1;
210
211     for (i = 6; i < 11; ++i) {
212         nowbin[i] = rs[i - 6];
213         nowbin[i + 5] = rt[i - 6];
214     }
215     for (i = 16; i < 32; ++i)
216         nowbin[i] = imm[i - 16];
217
218     nowhex = binToHex(nowbin);
219
220     for (i = 0; i < 8; ++i) {
221         outfile << nowhex[i];
222     }
223     outfile << ", ";
224     cnt++;
225 }
226 else if (p0 == "lui") {
227     nowbin[2] = nowbin[3] = nowbin[4] = nowbin[5] = 1;
228     iss >> p1;
229     strcpy(p2, p1.c_str());
230     int* rt = getReg(p2);
231     iss >> p1;
232     strcpy(p2, p1.c_str());
233     int* imm = numToBin(p2);
234     for (i = 11; i < 16; ++i)
235         nowbin[i] = rt[i - 11];
236     for (i = 16; i < 31; ++i)
237         nowbin[i] = imm[i - 16];
238
239     nowhex = binToHex(nowbin);
240
241     for (i = 0; i < 8; ++i) {
242         outfile << nowhex[i];
243     }

```

```

244     outfile << ", ";
245     cnt++;
246 }
247 else if (p0 == "sw" || p0 == "lw") {
248     iss >> p1;
249     strcpy(p2, p1.c_str());
250     int* rt = getReg(p2);
251     iss >> p1;
252     strcpy(p2, p1.c_str());
253     int* imm;
254     if (*p2 == '(')
255         imm = iToBin(0, 16);
256     else {
257         i = 0;
258         int t, reg = 0;
259         while (p2[i] != '(') {
260             t = p2[i] - '0';
261             reg = reg * 10 + t;
262             i++;
263         }
264         imm = iToBin(reg, 16);
265     }
266     iss >> p1;
267     strcpy(p2, p1.c_str());
268     int* rs = getReg(p2);
269     nowbin[0] = nowbin[4] = nowbin[5] = 1;
270     if (p0 == "sw") nowbin[2] = 1;
271     for (i = 6; i < 11; ++i) {
272         nowbin[i] = rs[i - 6];
273         nowbin[i + 5] = rt[i - 6];
274     }
275     for (i = 16; i < 32; ++i)
276         nowbin[i] = imm[i - 16];
277
278     nowhex = binToHex(nowbin);
279
280     for (i = 0; i < 8; ++i) {
281         outfile << nowhex[i];
282     }
283     outfile << ", ";
284     cnt++;
285 }
286 else if (p0 == "jr") {
287     iss >> p1;
288     strcpy(p2, p1.c_str());
289     int* rs = getReg(p2);
290     for (i = 6; i < 11; ++i)
291         nowbin[i] = rs[i - 6];
292     nowbin[28] = 1;
293
294     nowhex = binToHex(nowbin);
295
296     for (i = 0; i < 8; ++i) {
297         outfile << nowhex[i];
298     }
299     outfile << ", ";
300     cnt++;
301 }

```



```

302     else if (p0 == "j" || p0 == "jal") {
303         int offset;
304         for (j = 0; j <= structsize; ++j) {
305             if (jump[j].locate == line && jump[j].type == p0)
306                 break;
307         }
308         for (i = 0; i <= structsize; ++i) {
309             if (jump[i].Label == jump[j].Label && jump[i].type == "tab") {
310                 offset = jump[i].locate;
311                 offset *= 4;
312                 break;
313             }
314         }
315         int* addroff = iToBin(offset, 32); //Calculate pseudo absolute
address
316         nowbin[4] = 1;
317         if (p0 == "jal")
318             nowbin[5] = 1;
319         for (i = 6; i < 32; ++i)
320             nowbin[i] = addroff[i - 2];
321
322         nowhex = binToHex(nowbin);
323
324         for (i = 0; i < 8; ++i) {
325             outfile << nowhex[i];
326         }
327         outfile << ", ";
328         cnt++;
329     }
330     else { //Errors when entering illegal instructions
331         cout << "Error input!" << endl;
332         exit(1);
333     }
334     if (cnt % 8 == 0)
335         outfile << endl;
336 }
337
338 string mips::toLower(string s) {
339     int len = s.size();
340     for (int i = 0; i < len; i++) {
341         if (s[i] >= 'A' && s[i] <= 'Z') {
342             s[i] += ('a' - 'A');
343         }
344     }
345     return s;
346 }
347
348 int* mips::getReg(char* p) {
349     int reg = 0;
350     char* pt = p;
351
352     if (*pt == 'a') {
353         pt++;
354         if (*pt == 't')
355             reg = 1;
356         else {
357             reg = atoi(pt) + 4;
358         }

```

```

359     }
360     else if (*pt == 'v') {
361         pt++;
362         reg = atoi(pt) + 2;
363     }
364     else if (*pt == 't') {
365         pt++;
366         reg = atoi(pt);
367         if (reg <= 7)
368             reg += 8;
369         else reg += 16;
370     }
371     else if (*pt == 's') {
372         pt++;
373         if (*pt == 'p')
374             reg = 29;
375         else {
376             reg = atoi(pt) + 16;
377         }
378     }
379     else if (*pt == 'k') {
380         pt++;
381         reg = atoi(pt) + 26;
382     }
383     else if (*pt == 'z')
384         reg = 0;
385     else if (*pt == 'g')
386         reg = 28;
387     else if (*pt == 'f')
388         reg = 30;
389     else if (*pt == 'r')
390         reg = 31;
391     else {
392         cout << "Error input!" << endl;
393         exit(1);
394     }
395     int* q = iToBin(reg, 5);
396
397     return q;
398 }
399
400 int* mips::iToBin(int reg, int width) {
401     int* q = new int[width];
402     for (int i = width - 1; i >= 0; --i) {
403         q[i] = reg % 2;
404         reg = reg / 2;
405     }
406     return q;
407 }
408
409 int* mips::iToBinOff(int reg, int width) {
410     int sign = 0;
411     int i;
412     if (reg < 0) {
413         reg *= (-1);
414         sign = 1;
415     }
416     int* q = new int[width + 2];

```

```

417     for (i = width + 1; i >= 0; --i) {
418         q[i] = reg % 2;
419         reg = reg / 2;
420     }
421     int* qt = new int[width];
422     for (i = 0; i < width; ++i)
423         qt[i] = q[i];
424     if (sign) {
425         for (i = 0; i < width; ++i) {
426             if (qt[i]) qt[i] = 0;
427             else qt[i] = 1;
428         }
429         qt[width - 1]++;
430         for (i = width - 1; i >= 0; --i) {
431             if (qt[i] == 1)
432                 break;
433             qt[i] = 0;
434             if (i >= 1) {
435                 qt[i - 1]++;
436             }
437         }
438     }
439     return qt;
440 }
441
442 int* mips::numToBin(char* p) {
443     int i = 0;
444     int j = -1;
445     int k = 0;
446     int t;
447     int reg;
448     int* q = new int[16];
449     char* pt = p;
450     if (pt[1] == 'x' || pt[1] == 'X') { // If expressed in hexadecimal
451         // pt[2]~pt[5]
452         for (i = 2; i < 6; ++i) {
453             if (isupper(pt[i]))
454                 pt[i] += ('a' - 'A');
455             if (islower(pt[i]))
456                 t = pt[i] - 'a' + 10;
457             else if (isdigit(pt[i]))
458                 t = pt[i] - '0';
459             for (j = 4 * (i - 1) - 1; j >= 4 * (i - 2); --j) {
460                 q[j] = t % 2;
461                 t = t / 2;
462             }
463         }
464     }
465     else {
466         i = 0;
467         if (pt[0] == '-')
468             i++;
469         t = reg = 0;
470         k = 1;
471         while (pt[i] != '\0') {
472             t = pt[i] - '0';
473             reg = reg * 10 + t;
474             i++;

```

```

475         if (pt[i] == '/')
476             break;
477     }
478     q = iToBin(reg, 16);
479     if (*p != '-')
480         return q;
481     for (i = 0; i < 16; ++i) { //Negate
482         if (q[i]) q[i] = 0;
483         else q[i] = 1;
484     }
485     q[15]++;
486     for (i = 15; i >= 0; --i) { //+1
487         if (q[i] == 1) break;
488         if (q[i] == 2) {
489             q[i] = 0;
490             if (i != 0)
491                 q[i - 1]++;
492         }
493     }
494 }
495
496 return q;
497 }
498
499 char* mips::binToHex(int* b) {
500     int i, j;
501     int t, k, reg;
502     char* q = new char[8];
503     for (i = 0; i < 32; i += 4) {
504         t = reg = 0;
505         k = 1;
506         for (j = 3; j >= 0; --j)
507         {
508             t = k * b[i + j];
509             reg += t;
510             k *= 2;
511         }
512         if (reg <= 9)
513             q[i / 4] = reg + '0';
514         else
515             q[i / 4] = reg - 10 + 'A';
516     }
517     return q;
518 }

```

Invert.cpp

```

1  #include "Invert.h"
2  #include "lineProcess.h"
3
4  Invert::Invert(string path) {
5      ifstream infile(path);
6      ins tmp;
7      while (!infile.eof()) {
8          infile >> tmp.type >> tmp.name >> tmp.op;

```

```

9         if (tmp.type == "R")
10             R_type.push_back(tmp);
11         else if (tmp.type == "I")
12             I_type.push_back(tmp);
13         else
14             J_type.push_back(tmp);
15     }
16     infile.close();
17     sort(R_type.begin(), R_type.end(), cmp);
18     sort(I_type.begin(), I_type.end(), cmp);
19     sort(J_type.begin(), J_type.end(), cmp);
20 }
21
22 void Invert::Convert(string file1, string file2) {
23     ifstream infile(file1);
24     ofstream outfile(file2);
25     string str;
26     if (!infile) {
27         cout << "Open error!" << endl;
28         exit(1);
29     }
30     while (getline(infile, str)) {
31         lineProcess linep(str, ";");//Remove multiple spaces and tabs at
the beginning and end of lines
32         str = linep.Cut();
33         if (str.empty()) continue;//continue if line is empty
34         if (!(str[0] >= '0' && str[0] <= '9') && !(str[0] >= 'a' && str[0]
<= 'f')
35             && !(str[0] >= 'A' && str[0] <= 'F'))
36             continue;
37         istringstream iss(str);
38         string s;
39         while (iss >> s) { //Read each machine code in the line
40             string newline = hexToBin(s); //Hexadecimal machine code to
binary
41             string op = newline.substr(0, 6);
42             if (op == "000000") { //Determine the type according to the op
value
43                 string instructor_name = findName(R_type,
newline.substr(26, 6));
44                 string rs, rt, rd, shamt;
45                 string rss, rts, rds, shamts;
46                 int rsi, rti, rdi, shamti;
47                 rs = newline.substr(6, 5);
48                 rt = newline.substr(11, 5);
49                 rd = newline.substr(16, 5);
50                 shamt = newline.substr(21, 5);
51                 rsi = binToInt(rs);
52                 rti = binToInt(rt);
53                 rdi = binToInt(rd);
54                 outfile << instructor_name << " $";
55                 if (instructor_name == "sll" || instructor_name == "srl")
56                     outfile << iToSymbol(rdi) << ", $" << iToSymbol(rti) <<
", " << binToInt(shamt) << endl;
57                 else if (instructor_name == "jr")
58                     outfile << iToSymbol(rsi) << endl;
59                 else

```

```

60         outfile << iToSymbol(rdi) << ",$" << iToSymbol(rsi) <<
    "$" << iToSymbol(rti) << endl;
61     }
62     else if (op == "000010" || op == "000011") {
63         string instructor_name = findName(J_type, op);
64         string address = newline.substr(6, 26);
65         int addri = binToInt(address);
66         char* addrh;
67         addrh = iToHex(addri);
68         outfile << instructor_name << " 0x";
69         for (int i = 0; i < 8; ++i)
70             outfile << addrh[i];
71         outfile << endl;
72     }
73     else {
74         string instructor_name = findName(I_type, op);
75         string rs, rt, immediate;
76         int rsi, rti;
77         rs = newline.substr(6, 5);
78         rt = newline.substr(11, 5);
79         rsi = binToInt(rs);
80         rti = binToInt(rt);
81         immediate = newline.substr(16, 16);
82         outfile << instructor_name << " $";
83         if (instructor_name == "lui")
84             outfile << iToSymbol(rti) << "," <<
binToInt(immediate) << endl;
85         else if (instructor_name == "lw" || instructor_name ==
"sw")
86             outfile << iToSymbol(rti) << "," <<
binToInt(immediate) << "($" << binToInt(rs) << ")" << endl;
87         else if (instructor_name == "beq" || instructor_name ==
"bne")
88             outfile << iToSymbol(rsi) << ",$" << iToSymbol(rti) <<
", " << binToInt(immediate) << endl;
89         else
90             outfile << iToSymbol(rti) << ",$" << iToSymbol(rsi) <<
", " << binToInt(immediate) << endl;
91     }
92 }
93 }
94 }
95
96 bool Invert::cmp(ins a, ins b) {
97     return a.op < b.op;
98 }
99
100 string Invert::findName(const vector<ins>& v, string s) {
101     int start = 0, end = v.size() - 1;
102     while (start <= end) {
103         int mid = (start + end) / 2;
104         if (v[mid].op == s)
105             return v[mid].name;
106         else if (v[mid].op < s)
107             start = mid + 1;
108         else
109             end = mid - 1;
110     }

```

```

111     return "";
112 }
113
114 int Invert::binToInt(string s) {
115     int res = 0;
116     for (int i = 0; i < s.length(); ++i) {
117         res = res * 2 + (s[i] - '0');
118     }
119     return res;
120 }
121
122 string Invert::hexToBin(string h) {
123     int i, j;
124     int t;
125     int* b = new int[32];
126     char* bc = new char[32];
127     string bs;
128     for (i = 0; i < 8; ++i) {
129         if (isupper(h[i]))
130             h[i] += ('a' - 'A');
131         if (islower(h[i]))
132             t = h[i] - 'a' + 10;
133         else if (isdigit(h[i]))
134             t = h[i] - '0';
135         for (j = 4 * (i + 1) - 1; j >= 4 * i; --j) {
136             b[j] = t % 2;
137             t = t / 2;
138             bc[j] = b[j] + '0';
139         }
140     }
141     bs = bc;
142     return bs;
143 }
144
145 string Invert::iToSymbol(int t)
146 {
147     string name;
148     switch (t)
149     {
150     case(0):
151         name = "zero";
152         break;
153     case(1):
154         name = "at";
155         break;
156     case(2):
157         name = "v0";
158         break;
159     case(3):
160         name = "v1";
161         break;
162     case(4):
163         name = "a0";
164         break;
165     case(5):
166         name = "a1";
167         break;
168     case(6):

```

```
169         name = "a2";
170         break;
171     case(7):
172         name = "a3";
173         break;
174     case(8):
175         name = "t0";
176         break;
177     case(9):
178         name = "t1";
179         break;
180     case(10):
181         name = "t2";
182         break;
183     case(11):
184         name = "t3";
185         break;
186     case(12):
187         name = "t4";
188         break;
189     case(13):
190         name = "t5";
191         break;
192     case(14):
193         name = "t6";
194         break;
195     case(15):
196         name = "t7";
197         break;
198     case(16):
199         name = "s0";
200         break;
201     case(17):
202         name = "s1";
203         break;
204     case(18):
205         name = "s2";
206         break;
207     case(19):
208         name = "s3";
209         break;
210     case(20):
211         name = "s4";
212         break;
213     case(21):
214         name = "s5";
215         break;
216     case(22):
217         name = "s6";
218         break;
219     case(23):
220         name = "s7";
221         break;
222     case(24):
223         name = "t8";
224         break;
225     case(25):
226         name = "t9";
```



```

227         break;
228     case(26):
229         name = "k0";
230         break;
231     case(27):
232         name = "k1";
233         break;
234     case(28):
235         name = "gp";
236         break;
237     case(29):
238         name = "sp";
239         break;
240     case(30):
241         name = "fp";
242         break;
243     case(31):
244         name = "ra";
245         break;
246     default:
247         break;
248     }
249     return name;
250 }
251
252 char* Invert::iToHex(int reg) {
253     int* q = new int[32];
254     char* h = new char[8];
255     for (int i = 31; i >= 0; --i) {
256         q[i] = reg % 2;
257         reg = reg / 2;
258     }
259     h = binToHex(q);
260     return h;
261 }
262
263 char* Invert::binToHex(int* b) {
264     int i, j;
265     int t, k, reg;
266     char* q = new char[8];
267     for (i = 0; i < 32; i += 4) {
268         t = reg = 0;
269         k = 1;
270         for (j = 3; j >= 0; --j)
271         {
272             t = k * b[i + j];
273             reg += t;
274             k *= 2;
275         }
276         if (reg <= 9)
277             q[i / 4] = reg + '0';
278         else
279             q[i / 4] = reg - 10 + 'A';
280     }
281     return q;
282 }

```

lineprocess.cpp

```
1  #include "lineProcess.h"
2
3  lineProcess::lineProcess(string& s, string c)
4  {
5      str = s;
6      comment_str = c;
7  };
8
9  string lineProcess::Cut()
10 {
11     for (char& c : str)
12     {
13         //Unify into spaces
14         if (c == '\t' || c == ',' || c == ';' || c == '\r' || c == '\n' ||
15             c == '$')
16             c = ' ';
17     }
18
19     str.erase(0, str.find_first_not_of(" ")); //Remove head-of-line spaces
20     str.erase(str.find_last_not_of(" ") + 1); //Remove end-of-line spaces
21
22     //Find the position of the comment character, if it does not exist, get
23     string ::npos
24     int n_comment_start = str.find_first_of(comment_str);
25     if (n_comment_start != string::npos)
26         str.erase(n_comment_start); //Delete comment
27
28     return str;
29 }
```

三、实验过程和数据记录及结果分析

1. Assembler

When we input 1, run the assembler. Then we input the name of the file we want to open.

We can find the answer in the file "mips.coe".

```
Choose assembler or dissembler, 1 is assembler and 0 is dissembler
1
Please input the name of the file you want to open.
scpu_demo_2018.asm
You can find the answer in the file "mips.coe".
```

the output is as follows

```

1 memory_initialization_radix=16;
2 memory_initialization_vector =
3 08000008, 00000020, 00000020, 00000020, 00000020, 00000020, 00000020,
4 00000020,
5 3C03F000, 00032040, 3C088000, 2014003F, 00083102, 00C33020, 00000827,
6 0020102A,
7 202AFFFF, AC660004, 8C650000, 00A52820, 00A52820, AC650000, 21290001,
8 AC890000,
9 8C0D0014, 8C650000, 00A52820, 00A52820, AC650000, 8C650000, 00A85824,
10 21AD0001,
11 15A00001, 0C000038, 8C650000, 20120008, 0252B020, 02569020, 00B25824,
12 11600005,
13 11720009, 20120008, 1172000A, AC890000, 08000019, 15410002, 00005027,
14 014A5020,
15 AC8A0000, 08000019, 8E290060, AC890000, 08000019, 8E290020, AC890000,
16 08000019,
17 8C0D0014, 014A5020, AD4A0001, 22310004, 02348824, 01224820, 15210001,
18 21290005,
19 8C650000, 00A55820, 016B5820, AC6B0000, AC660004, 03E00008,

```

2. Dissembler

When we input 0, run the dissembler. Then we input the name of the file we want to open.

We can find the answer in the file "invert.txt".

```

Choose assembler or dissembler, 1 is assembler and 0 is dissembler
0
Please input the name of the file you want to open.
scpu_demo_2018.coe
You can find the answer in the file "invert.txt".

```

the output is as follows.

```

1 j 0x00000008
2 add $zero,$zero,$zero
3 add $zero,$zero,$zero
4 add $zero,$zero,$zero
5 add $zero,$zero,$zero
6 add $zero,$zero,$zero
7 add $zero,$zero,$zero
8 add $zero,$zero,$zero
9 lui $v1,61440
10 sll $a0,$v1,1
11 lui $t0,32768
12 addi $s4,$zero,63
13 srl $a2,$t0,4
14 add $a2,$a2,$v1
15 nor $at,$zero,$zero
16 slt $v0,$at,$zero
17 addi $t2,$at,65535
18 sw $a2,4($3)
19 lw $a1,0($3)
20 add $a1,$a1,$a1
21 add $a1,$a1,$a1
22 sw $a1,0($3)

```

```

23  addi $t1,$t1,1
24  sw $t1,0($4)
25  lw $t5,20($0)
26  lw $a1,0($3)
27  add $a1,$a1,$a1
28  add $a1,$a1,$a1
29  sw $a1,0($3)
30  lw $a1,0($3)
31  and $t3,$a1,$t0
32  addi $t5,$t5,1
33  bne $t5,$zero,1
34  jal 0x00000038
35  lw $a1,0($3)
36  addi $s2,$zero,8
37  add $s6,$s2,$s2
38  add $s2,$s2,$s6
39  and $t3,$a1,$s2
40  beq $t3,$zero,5
41  beq $t3,$s2,9
42  addi $s2,$zero,8
43  beq $t3,$s2,10
44  sw $t1,0($4)
45  j 0x00000019
46  bne $t2,$at,2
47  nor $t2,$zero,$zero
48  add $t2,$t2,$t2
49  sw $t2,0($4)
50  j 0x00000019
51  lw $t1,96($17)
52  sw $t1,0($4)
53  j 0x00000019
54  lw $t1,32($17)
55  sw $t1,0($4)
56  j 0x00000019
57  lw $t5,20($0)
58  add $t2,$t2,$t2
59  ori $t2,$t2,1
60  addi $s1,$s1,4
61  and $s1,$s1,$s4
62  add $t1,$t1,$v0
63  bne $t1,$at,1
64  addi $t1,$t1,5
65  lw $a1,0($3)
66  add $t3,$a1,$a1
67  add $t3,$t3,$t3
68  sw $t3,0($3)
69  sw $a2,4($3)
70  jr $ra
71

```

四、讨论与心得

1. Through this assignment, I have a deeper understanding of the address calculation of the mips j-type instruction and the two jump instructions of beq and bne. I didn't know how to implement these two types of instructions at the beginning. With the help of the teaching

assistant, I chose to traverse the file twice. When traversing for the first time all the jump instructions and the addresses of the jump instructions that need to be read. When reading the file for the second time, calculate the required value based on the relative address.

2. For the disassembler, in order to facilitate the conversion of instructions into machine code, first write all instruction names and unique opcodes in the instruction.txt file, read the file to create three queues, and sort. When the mips command is output later, the corresponding command name can be obtained according to the command operation code.