

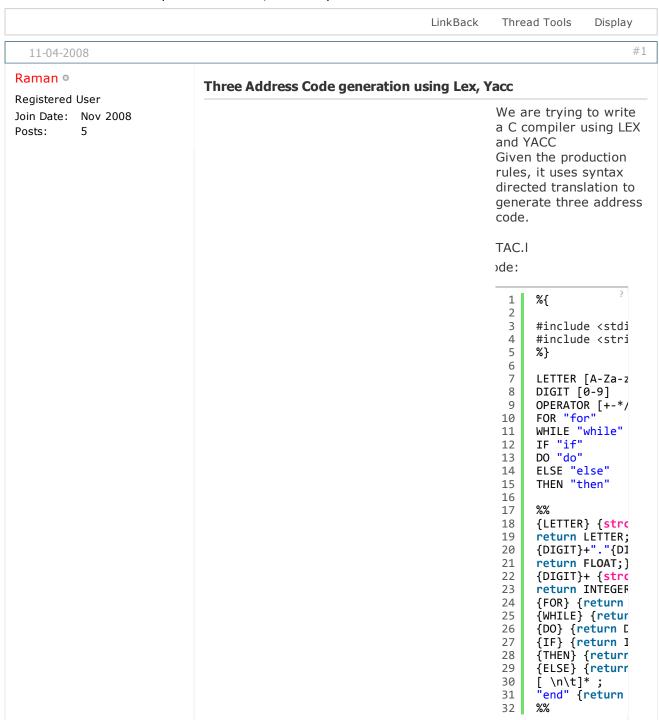
♠ Forum General Programming Boards C Programming

Three Address Code generation using Lex, Yacc

Getting started with C or C++ | C Tutorial | C++ Tutorial | C and C++ FAQ | Get a compiler | Fixes for common problems

Three Address Code generation using Lex, Yacc

This is a discussion on Three Address Code generation using Lex, Yacc within the **C Programming** forums, part of the General Programming Boards category; We are trying to write a C compiler using LEX and YACC Given the production rules, it uses syntax directed ...



TAC.y Code:

```
1
     %{
 2
 3
     #include<stdio.h>
 4
 5
     #include<stdlib.h>
 6
 7
     #include<string.h>
 8
     #include<ctype.h>
9
10
11
     #include<math.h>
12
13
     #define START 100
14
15
     typedef struct s
16
17
18
19
     int* true;
20
21
     int* false;
22
23
     int* next;
24
25
     int quad;
26
27
     char place[5];
28
29
     }ETYPE;
30
     int nextquad=START;
31
32
33
     char code[25][50];
34
35
     %}
36
37
38
39
     %union
40
41
42
     char id[10];
43
44
     ETYPE eval;
45
46
47
     }
48
49
50
     %left "|"
51
52
     %left "&"
53
54
55
     %left "!"
56
     %left "<" ">"
57
58
     %left "+" "-"
59
60
     %left "*" "/"
61
62
63
     %left "(" ")"
64
65
66
67
68
     %right "="
69
70
71
     %token <id> LETTER INTEGER FLOAT
```

```
73
      %token FOR WHILE DO IF THEN ELSE
74
75
      %type <eval> PROGRAM BLOCK STATEMENTS ASSIGN COND L\
 76
77
 78
79
      %start PROGRAM
80
81
82
      %%
83
84
85
86
      PROGRAM: BLOCK {
87
88
89
      int i;
90
91
      Backpatch($1.next,nextquad);
92
93
      for(i=START;i<nextquad;i++)</pre>
94
95
      printf("\n%s",code[i-START]);
96
97
      printf("\n%d\n", nextquad);
98
99
      }
100
101
102
103
      BLOCK: DO M BLOCK WHILE M COND {
104
105
      Backpatch($6.true,$2.quad);
106
107
      $$.next=$6.false;
108
109
110
      | FOR ASSIGN M COND M ASSIGN Q BLOCK {
111
112
113
      Backpatch($8.next,$5.quad);
114
115
      Backpatch($6.next,$3.quad);
116
117
      Backpatch($4.true,$7.quad);
118
119
      $$.next=$4.false;
120
121
      sprintf(code[nextquad-START],"%d\tgoto %d",nextquad.
122
123
      Gen();
124
125
126
127
      | WHILE M COND M BLOCK {
128
129
      Backpatch($5.next,$2.quad);
130
131
      Backpatch($3.true,$4.quad);
132
133
      $$.next=$3.false;
134
135
      sprintf(code[nextquad-START],"%d\tgoto %d",nextquad.
136
      Gen();
137
138
139
140
      | IF COND M BLOCK {
141
142
      Backpatch($2.true,$3.quad);
143
144
145
      $$.next=Merge($2.false,$4.next);
146
147
148
149
      | IF COND M BLOCK N ELSE M BLOCK {
150
```

```
151
      Backpatch($2.true,$3.quad);
152
153
      Backpatch($2.false,$7.quad);
154
155
      $$.next=Merge($4.next,$5.next);
156
157
      $$.next=Merge($$.next,$8.next);
158
159
      }
160
      | '{' STATEMENTS '}' {
161
162
163
      $$.next=$2.next;
164
165
166
      | ASSIGN ';' {
167
168
      $$.next=(int*)malloc(sizeof(int)*15);
169
170
      $$.next[0]=0;
171
172
173
174
175
      | E { }
176
177
178
      STATEMENTS: STATEMENTS M BLOCK {
179
180
      Backpatch($1.next,$2.quad);
181
182
      $$.next=$3.next;
183
184
185
      }
186
      | BLOCK {
187
188
      $$.next=$1.next;
189
190
191
      }
192
193
194
195
      ASSIGN: LVAL '=' E {
196
      sprintf(code[nextquad-START],"%d\t%s = %s",nextquad.
197
198
      Gen();
199
200
201
      }
202
203
      | E { }
204
205
206
      LVAL: LETTER {strcpy($$.place,$1);}
207
208
209
210
211
      E: E '+' E {
212
213
      strcpy($$.place,Newtemp());
214
215
      sprintf(code[nextquad-START],"%d\t%s = %s + %s",next
216
217
      Gen();
218
219
      }
220
221
      | E '-' E {
222
223
      strcpy($$.place,Newtemp());
224
225
      sprintf(code[nextquad-START],"%d\t%s = %s - %s",next
226
227
      Gen();
228
229
```

```
230
      | E '*' E {
231
232
233
      strcpy($$.place,Newtemp());
234
      sprintf(code[nextquad-START],"%d\t%s = %s * %s",next
235
236
      Gen();
237
238
239
      }
240
      | E '/' E {
241
242
      strcpy($$.place,Newtemp());
243
244
      sprintf(code[nextquad-START],"%d\t%s = %s / %s",next
245
246
247
      Gen();
248
249
      }
250
      | '-' E %prec '*' {
251
252
253
      strcpy($$.place,Newtemp());
254
255
      sprintf(code[nextquad-START],"%d\t%s = - %s",nextquad-
256
257
      Gen();
258
259
      }
260
261
      | LETTER {
262
263
      strcpy($$.place,$1);
264
265
      }
266
      | INTEGER {
267
268
      strcpy($$.place,$1);
269
270
271
      }
272
273
      | FLOAT {
274
      strcpy($$.place,$1);
275
276
277
      }
278
279
280
      COND: COND '&' M COND {
281
282
283
      Backpatch($1.true,$3.quad);
284
285
      $$.true=$4.true;
286
287
      $$.false=Merge($1.false,$4.false);
288
289
290
      COND ' M COND {
291
292
293
      Backpatch($1.false,$3.quad);
294
295
      $$.true=Merge($1.true,$4.true);
296
      $$.false=$4.false;
297
298
299
300
      | '!' COND {
301
302
      $$.true=$2.false;
303
304
305
      $$.false=$2.true;
306
307
      }
```

```
309
      | '(' COND ')' {
310
311
      $$.true=$2.true;
312
      $$.false=$2.false;
313
314
315
      }
316
      | E '<' E {
317
318
319
      $$.true=Makelist(nextquad);
320
321
      $$.false=Makelist(nextquad+1);
322
323
      sprintf(code[nextquad-START],"%d\tif %s < %s goto ".</pre>
324
325
      Gen();
326
327
      sprintf(code[nextquad-START],"%d\tgoto ",nextquad);
328
329
      Gen();
330
331
332
      | E '>' E {
333
334
335
      $$.true=Makelist(nextquad);
336
337
      $$.false=Makelist(nextquad+1);
338
      sprintf(code[nextquad-START],"%d\tif %s > %s goto ".
339
340
341
      Gen();
342
343
      sprintf(code[nextquad-START],"%d\tgoto ",nextquad);
344
345
      Gen();
346
347
348
      | E {
349
350
      $$.true=Makelist(nextquad);
351
352
353
      $$.false=Makelist(nextquad+1);
354
      sprintf(code[nextquad-START],"%d\tif %s goto ",next
355
356
357
      Gen();
358
359
      sprintf(code[nextquad-START],"%d\tgoto ",nextquad);
360
361
      Gen();
362
363
      }
364
365
366
367
      M: {
368
369
      $$.quad=nextquad;
370
371
372
373
      N: {
374
375
      $$.next=Makelist(nextquad);
376
377
      sprintf(code[nextquad-START],"%d\tgoto ",nextquad);
378
379
      Gen();
380
381
      }
382
383
      Q: {
384
385
      $$.next=Makelist(nextquad);
386
387
      sprintf(code[nextquad-START],"%d\tgoto ",nextquad);
```

```
388
389
      Gen();
390
391
      $$.quad=nextquad;
392
393
394
395
396
      %%
397
398
399
400
401
      #include "lex.yy.c"
402
403
404
405
406
      main()
407
408
      {
409
410
      yyparse();
411
412
      }
413
      */
414
415
416
      yyerror(char *errmesg)
417
418
419
      printf("\n%s\n",errmesg);
420
421
422
      }
423
424
425
426
      char* Newtemp()
427
428
      {
429
430
      static int count=1;
431
432
      char* ch=(char*)malloc(sizeof(char)*5);
433
434
      sprintf(ch, "T%d", count++);
435
      return ch;
436
437
438
      }
439
440
441
      int* Makelist(int nquad)
442
443
444
      {
445
      int* list=(int*)malloc(sizeof(int)*15);
446
447
448
      list[0]=nquad;
449
450
      list[1]=0;
451
452
      return list;
453
454
      }
455
456
457
458
      int* Merge(int* list1,int* list2)
459
460
      {
461
462
      int* temp=list1,count1=0,count2=0;
463
464
      while(list1[count1]!=0) count1++;
465
466
      while(list1[count2]!=0)
```

```
467
      468
      469
            list1[count1]=list2[count2];
      470
      471
      472
             count1++;
      473
      474
             count2++;
      475
      476
             }
      477
      478
             return temp;
      479
      480
             }
      481
      482
      483
      484
             void Backpatch(int* list,int nquad)
      485
      486
             {
      487
             char addr[10];
      488
      489
             sprintf(addr,"%d",nquad);
      490
      491
      492
            while(*list!=0)
      493
      494
             {
      495
      496
             int index=*list++-START;
      497
      498
            strcat(code[index],addr);
      499
      500
            }
      501
      502
            }
      503
      504
      505
      506
            void Gen()
      507
      508
             {
      509
      510
            nextquad++;
      511
      512
            }
main.c
   Code:
          #include "y.tab.c"
#include "lex.yy.c"
      1
      2
      3
      4
          int main() {
      5
          yyparse();
      6
          return 0;
      7
lex TAC.I
yacc TAC.y
gcc main.c -II
./a.out
while (a < b) { a=a+b; c=c+b; }
Segmentation Fault
Can someone please help out?
```

#2

C_ntua •

Registered User

(the code is too big. Indent at least. Use debugging in order to find a segmentation fault. At least find out where it crashes or try different input and see what happens. Or wait for a yacc expert)

Join Date: Jun 2008
Posts: 1,853

« Previous Thread | Next Thread »

Popular pages

- Exactly how to get started with C++ (or C) today
- C Tutorial
- C++ Tutorial
- 5 ways you can learn to program faster
- The 5 Most Common Problems New Programmers Face
- How to set up a compiler
- 8 Common programming Mistakes
- What is C++11?
- · Creating a game, from start to finish

Recent additions

- How to create a shared library on Linux with GCC December 30, 2011
- Enum classes and nullptr in C++11 November 27, 2011
- Learn about The Hash Table November 20, 2011
- Rvalue References and Move Semantics in C++11 November 13, 2011
- C and C++ for Java Programmers November 5, 2011
- A Gentle Introduction to C++ IO Streams October 10, 2011

Similar Threads	
Enforcing Machine Code Restrictions? By SMurf in forum Tech Board	Replies: 21 Last Post: 03-30-2009, 07:34 AM
Help problem in winsock code	Replies: 8
By lolguy in forum C Programming	Last Post: 02-12-2009, 06:33 PM
I thought pointers were pointers By keira in forum C Programming	Replies: 19 Last Post: 08-15-2007, 11:48 PM
using YACC and lex to make your own shell	Replies: 2
By YankeePride13 in forum Linux Programming	Last Post: 12-28-2005, 09:00 AM
pointers	Replies: 13
By InvariantLoop in forum C Programming	Last Post: 02-04-2005, 08:32 AM



All times are GMT -6. The time now is 01:52 PM.

Powered by vBulletin® Version 4.2.3 Copyright © 2016 vBulletin Solutions, Inc. All rights reserved. Search Engine Optimization by vBSEO 3.6.1

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21