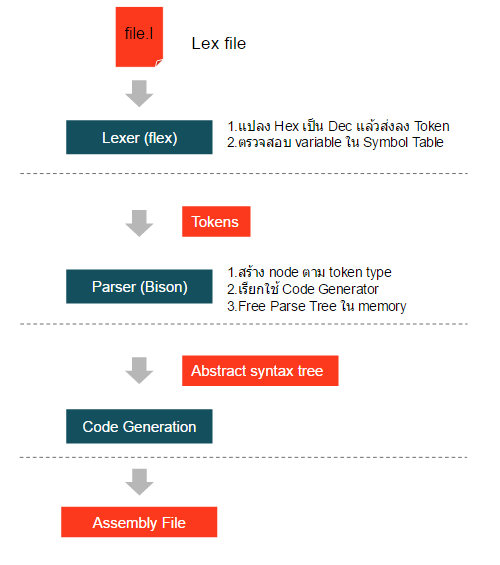
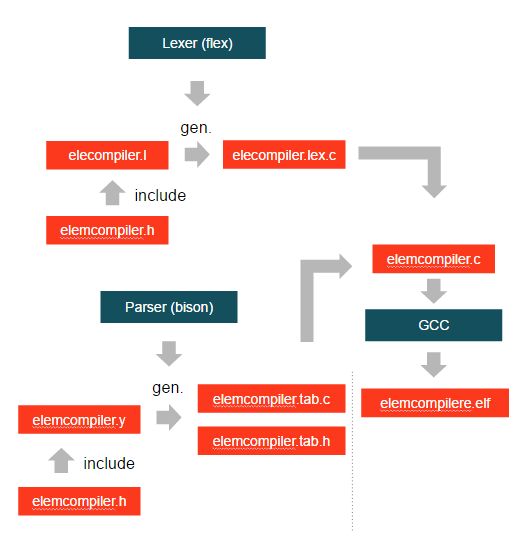
**1.แนวคิดและวิธีการ**



*รูปที่ 1 : ภาพรวมการทำงานของโปรเเกรม*



*รูปที่ 2 : ภาพรวมการทำงานของโปรเเกรม*

**File elecompiler.l**

ทำหน้าที่เป็น lexical analyzer หรือ lexer ในการเเบ่ง string ใน file input เพื่อแบ่งข้อมูล input ออกเป็นส่วน โดยไม่คิดความหมาย เรียกว่า token เพื่อนำไปตรวจสอบความหมายต่อไป โดยรูปแบบของ token แต่ละประเภทจะโดนเเยกด้วย regular expression (regex) โดยภายใน elecompiler.l จะมีการ include elecompiler.h เเละทำงานหลักๆคือ

* เเปลง string โดนใช้ regex เป็น token
* มีการเเปลง hex เป็น dec ก่อนเก็บลง token
* มีการ check symbol table ของ variable

โดยเมื่อนำ elecompiler.l ไปผ่าน flex เเล้วจะได้ไฟล์ elecomiler.lex.c ที่เป็นไฟล์ code ที่ได้จากการ gen ของ flex

**File elecompiler.y**

ทำหน้าที่เป็น Syntax Analyzer หรือ parser เพื่อตรวจสอบไวยากรณ์หรือความหมายของ input เนื่องจาก flex ไม่ได้ตรวจสอบความหมายเเต่เพียงตรวจสอบว่าคำถูกต้องหรือไม่ ภายใน parser จะทำการเเยก syntex ตามที่ได้ กำหนดไว้ โดยมีลำดับการทำงานคือ

* 1.ทำการ parse จากล่างขึ้นบน (bottom-up)
* 2.สร้าง node ของเเต่ละ node ขึ้นมาตาม type ของ node เเต่ละ node
* 3.เมื่อถึง root node ก็จะทำการ  return Abstract syntax tree ไปที่ generate assembly โดย
* 4.ทำการ free parse tree ใน memory ออก

**File elecompiler.h**

เป็นส่วนที่ใช้ในการกำหน้าเเละสร้าง โครงสร้างของ node เเต่ละ type โดยจะมี attribute ของ node หลักๆ คือ left\_node , right\_node ที่จะบอกตำเเหน่งของ node ต่อไปซึ่งเป็นตัวกระทำหรือตัวดำเนินการทางด้านซ้ายด้านขวาเเละ value ซึ่งจะเก็บค่าของ token ของ node นั้นๆ

**FIle elecompiler.c**

เก็บ function หลักๆที่ใช้ในการสร้าง assembly code โดยจะทำการ tarverse ที่ใช้สร้าง code ภาษา assembly เเละทำการตรวงสอบว่า เเต่ละ node มีตัวกระทำเป็นอะไรเเละทำการ เขียน assembly code ตามตัวกระทำนั้นๆ

**FIle elecompiler.tab.c เเละ elecompiler.tab.h**

เป็นไฟล์ที่ได้จากการ gen ของ bison เพื่อให้สามารถนำไปใช้งานได้โดยไ่ม่จำเป็นที่ต้องมี bison เเละ elecompiler.y

**File elecompiler.lex.c**

เป็นไฟล์ที่ได้จากการ gen ของ flex เพื่อให้สามารถนำไปใช้งานได้โดยไ่ม่จำเป็นที่ต้องมี flexเเละ elecompiler.l

**FIle Makefile**  
 เป็นไฟล์ทีใช้ในการช่วยการ gen ไฟล์ flex , bison เเละ assembly  โดยมีการทำงานดังนี้

ขั้นตอนการสร้าง ไฟล์

* 1. run คำสั่ง “bison -d elemcompiler.y” เพื่อ gen file elecompiler.tab.h เเละ  elecompiler.tab.c
* 2. run คำสั่ง “flex -o elecompiler.lex.c elecompiler.l” เพื่อ gen file elecompiler.lex.c
* 3. run คำสั่ง “gcc -Wall elecompiler.c elecompiler.tab.c elecompiler.lex.c -o $@ -lm”

**2.ไวยากรณ์และตัวอย่างภาษา**

**1. การ Assign ค่าตัวแปร**

Bison : VARIABLE\_NAME = EXPRESSION ;

Example : a = 1;

**2. เงื่อนไข (If statement)**

Bison : IF ( EXPRESSION ) { EXPRESSION\_LIST }

Example  : if (a) {

         a = 1+4;

         b = 2;

              }

**3. การวนลูป (Loop statement)**

Bison : loop ( NUM\_1 -> NUM\_2 , NUM\_3 ) { EXPRESSION\_LIST }

Example : loop ( 0 -> 10 , 1 ) {

                  a = 2\*3;

                  a = (2+2)\*3;

   }

**4. การแสดงค่า Print**

Bison-String : PRINT ( STRING ) ;

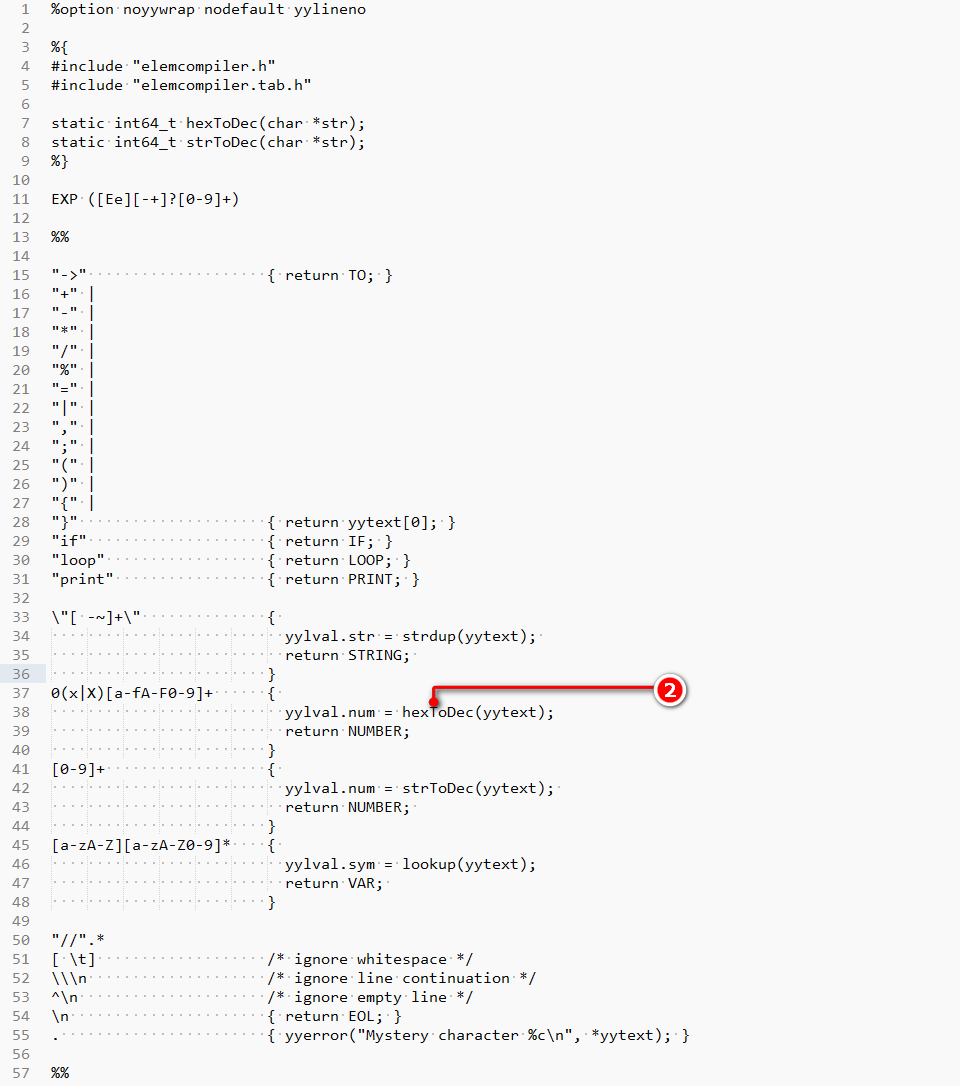
Example : print ( “Hello Compil…..err” ) ;

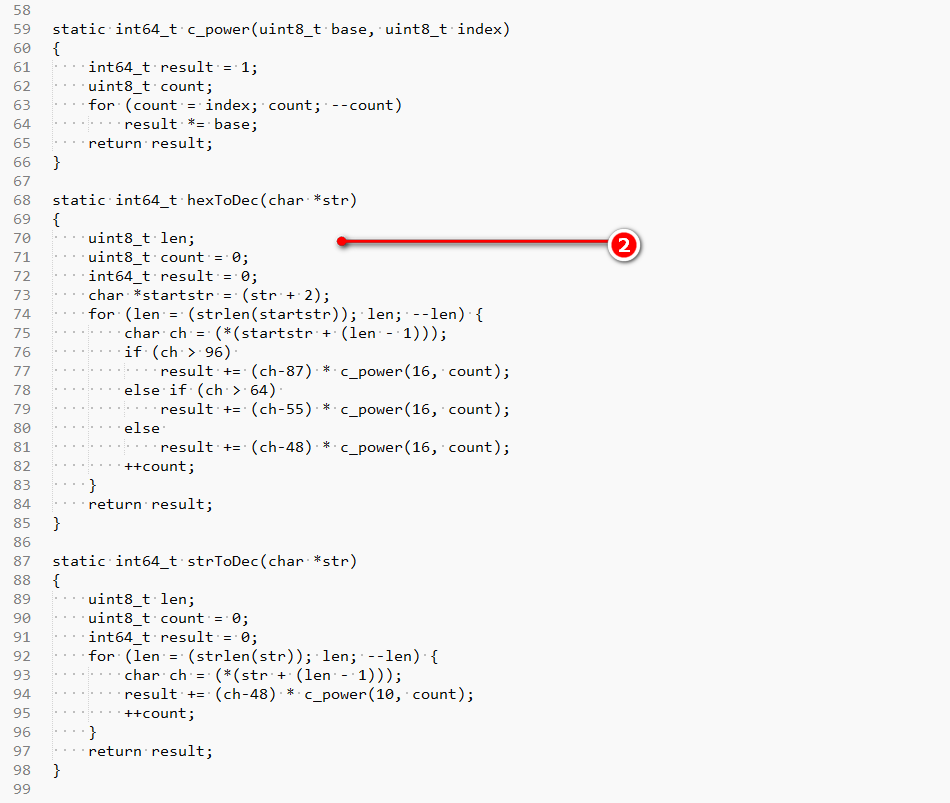
Bison-Expression : PRINT ( EXPRESSION ) ;

Example : print ( 1+1 );

**3.คำอธิบาย Source code**

**Flex File: elecompiler.l**

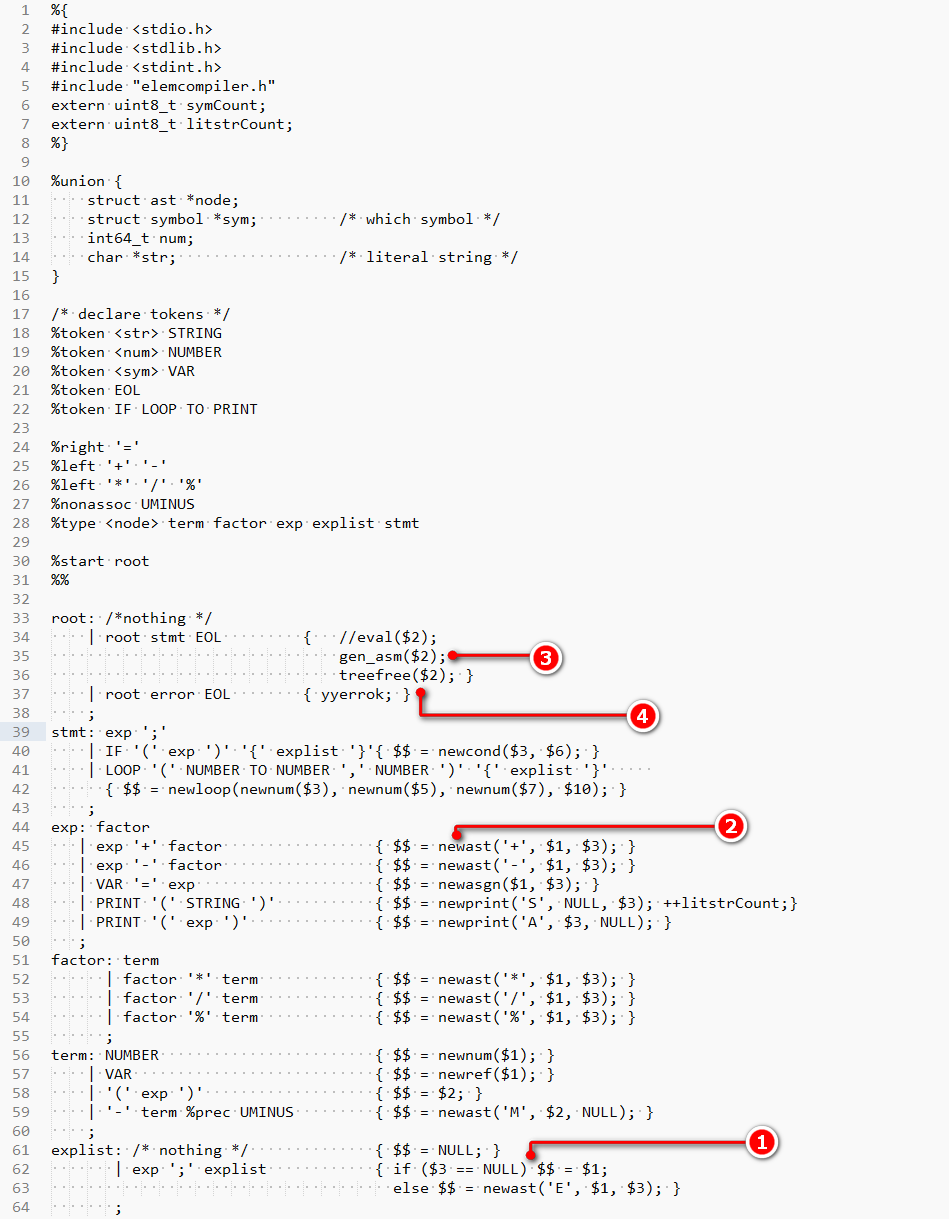




**อธิบายหลักการและส่วนหลักๆของไฟล์ elecompiler.l**

* (1) ส่วนล่างสุดของ File elecompiler.l เป็นส่วนของ User function ที่ถูกสร้างขึ้นและประกาศใช้โดนเป็น function ทีใช้ในการ แปลงเลข HEX เป็น DEC
* (2) ทำการแปลง regular expression ที่ใด้จาก HEX เป็น DEC โดยใช้ User Function ที่ประกาศไว้แล้วเก็บค่าที่เป็น DEC ลง token

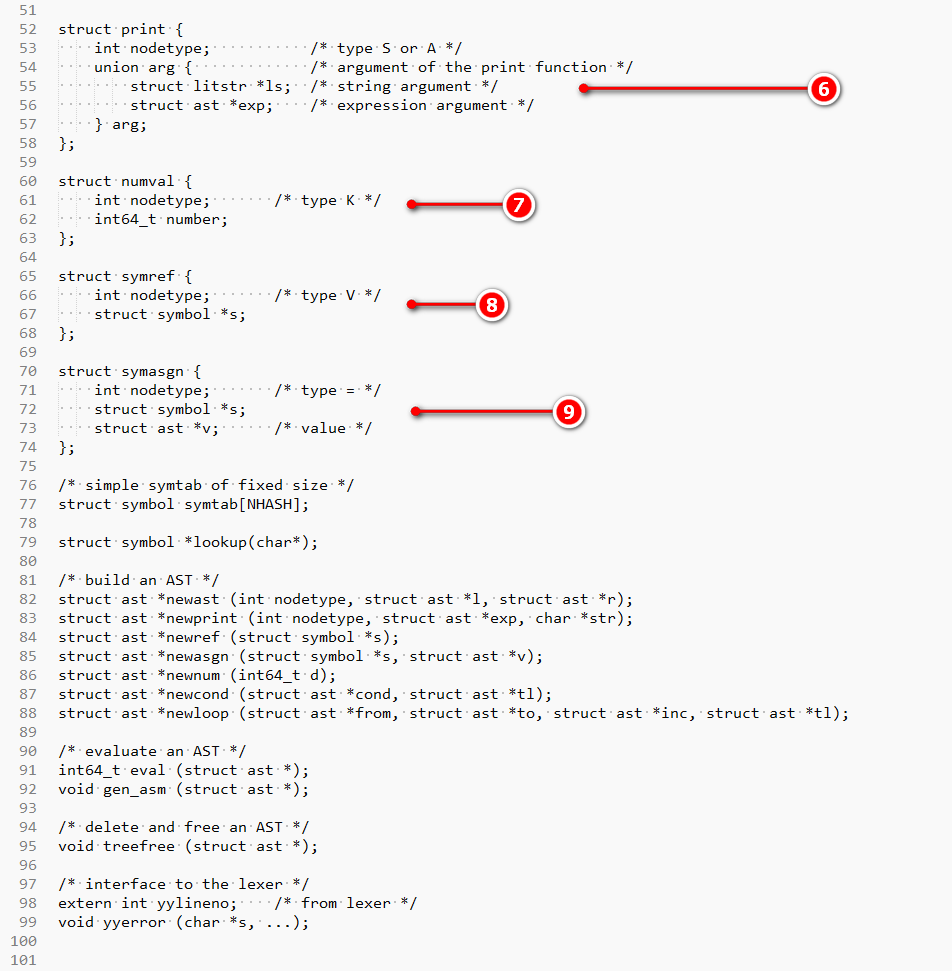
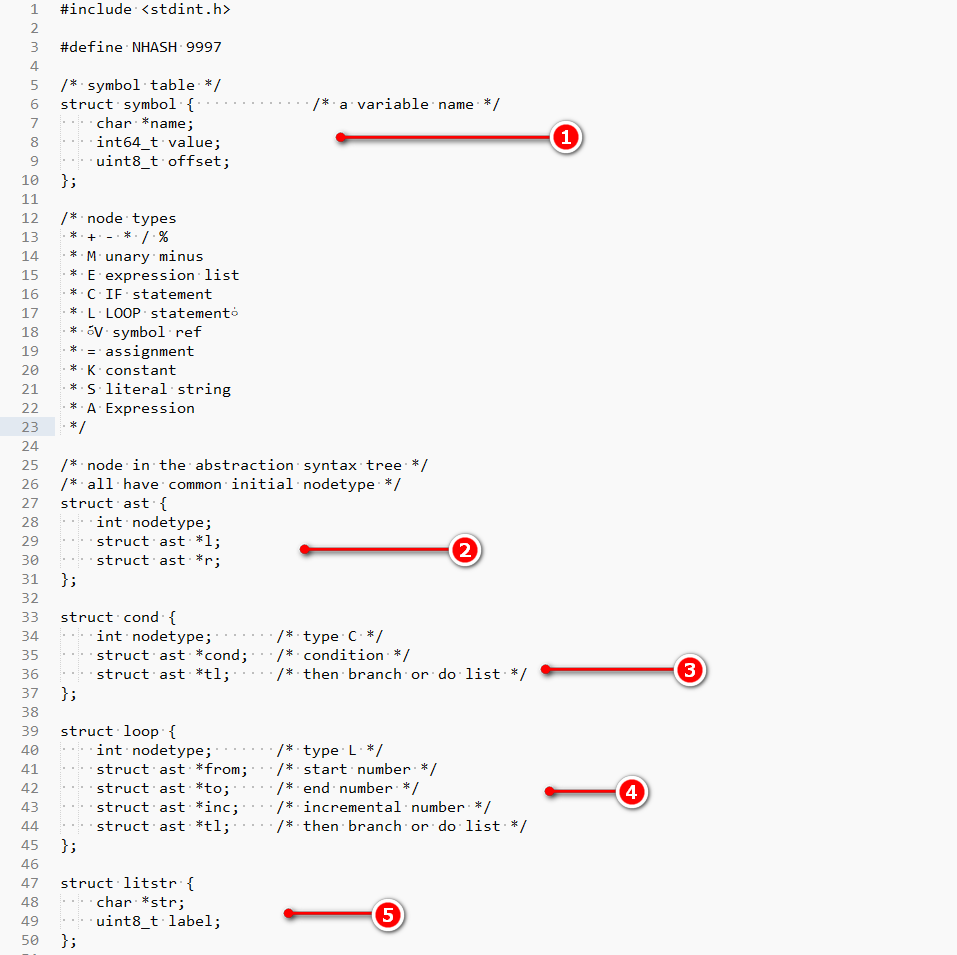
**Bison File : elecompiler.y**



**อธิบายหลักการและส่วนหลักๆของไฟล์ elecompiler.y**

* (1,2) ส่วนล่างสุดของ Abstract syntax tree ที่โดนสร้างขึ้นมา
* (3) เมื่อสร้างมาจนถึง root ของ Abstract syntax tree ก็จะทำการเรียก function gen\_ast() เพื่อทำการ generate assembly code ต่อไป
* (4) หลังจากทำการ generate assembly code ทำการ free memory ที่เก็บ Abstract syntax tree

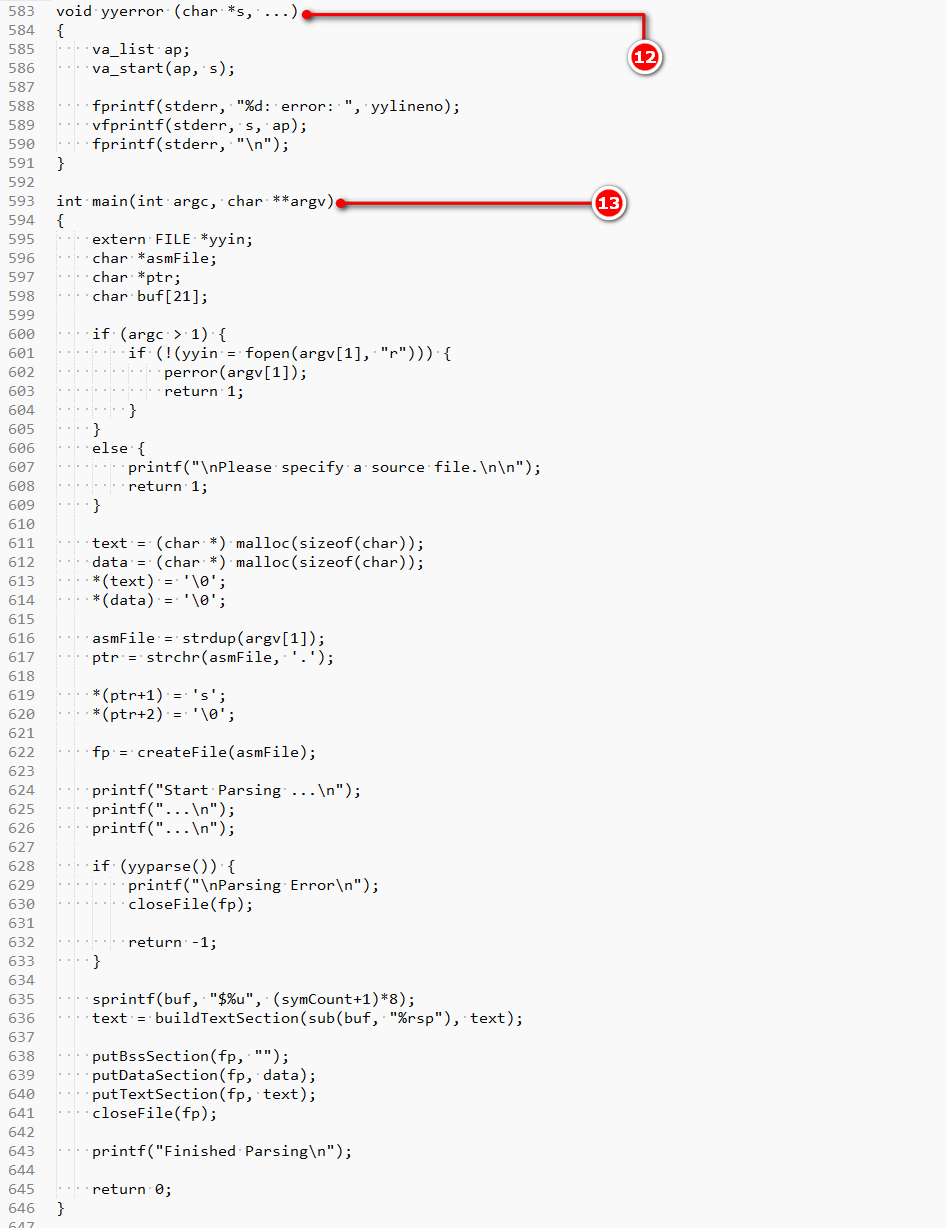
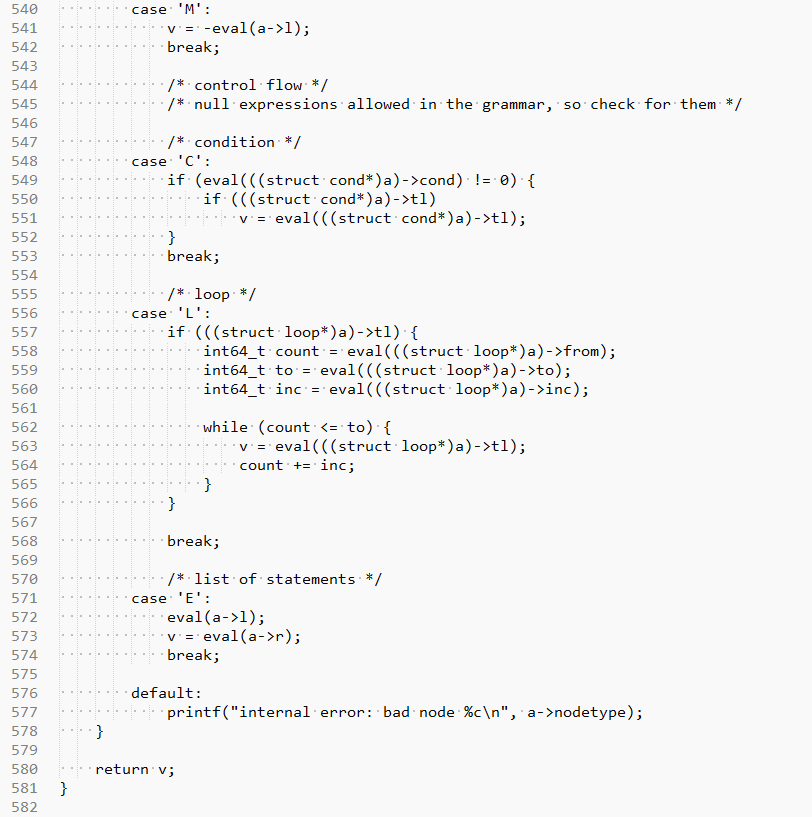
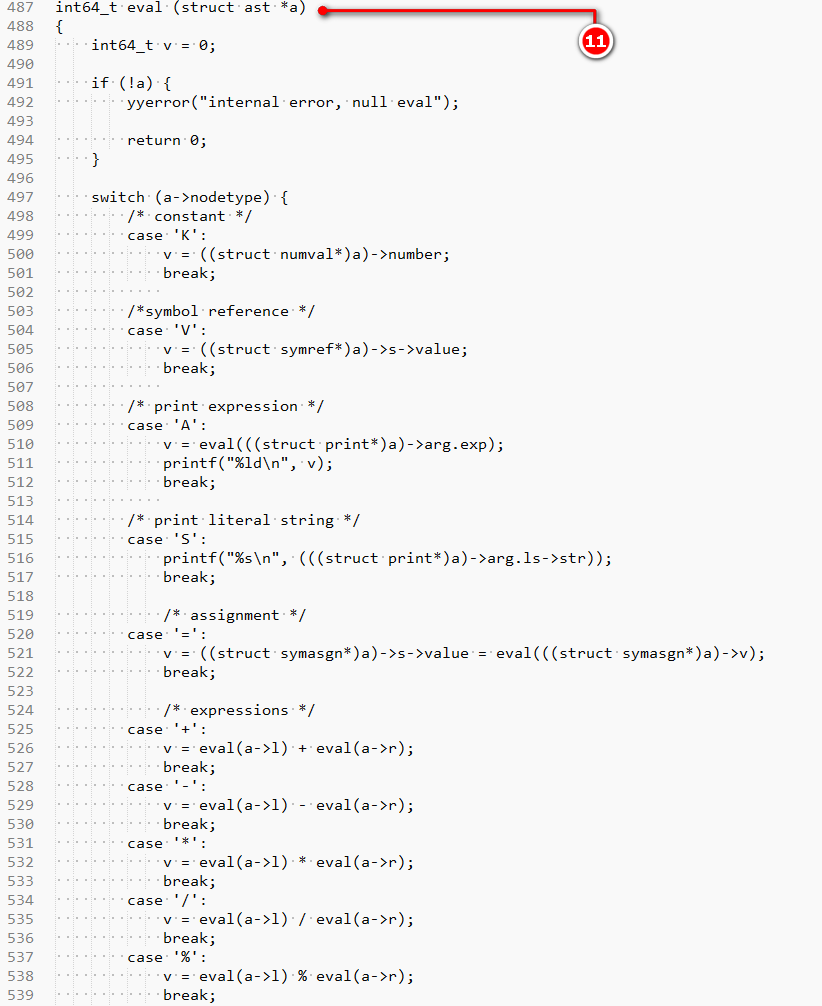
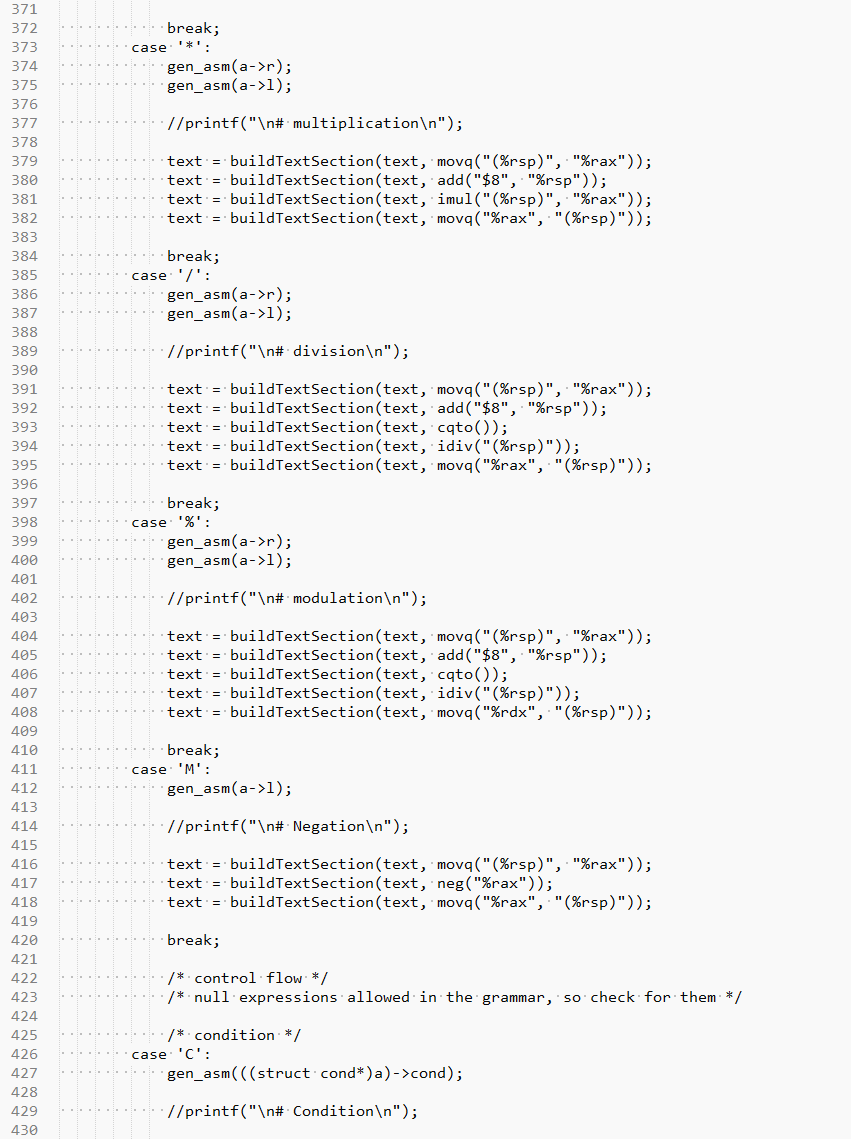
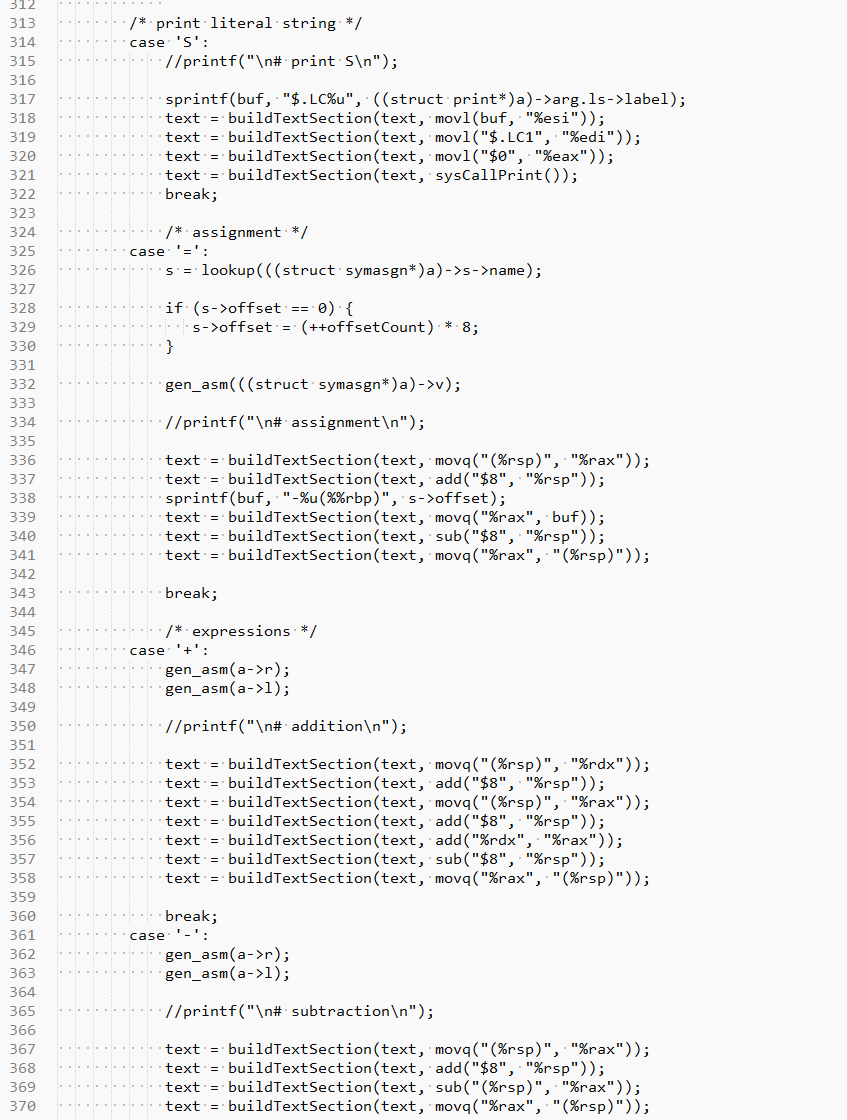
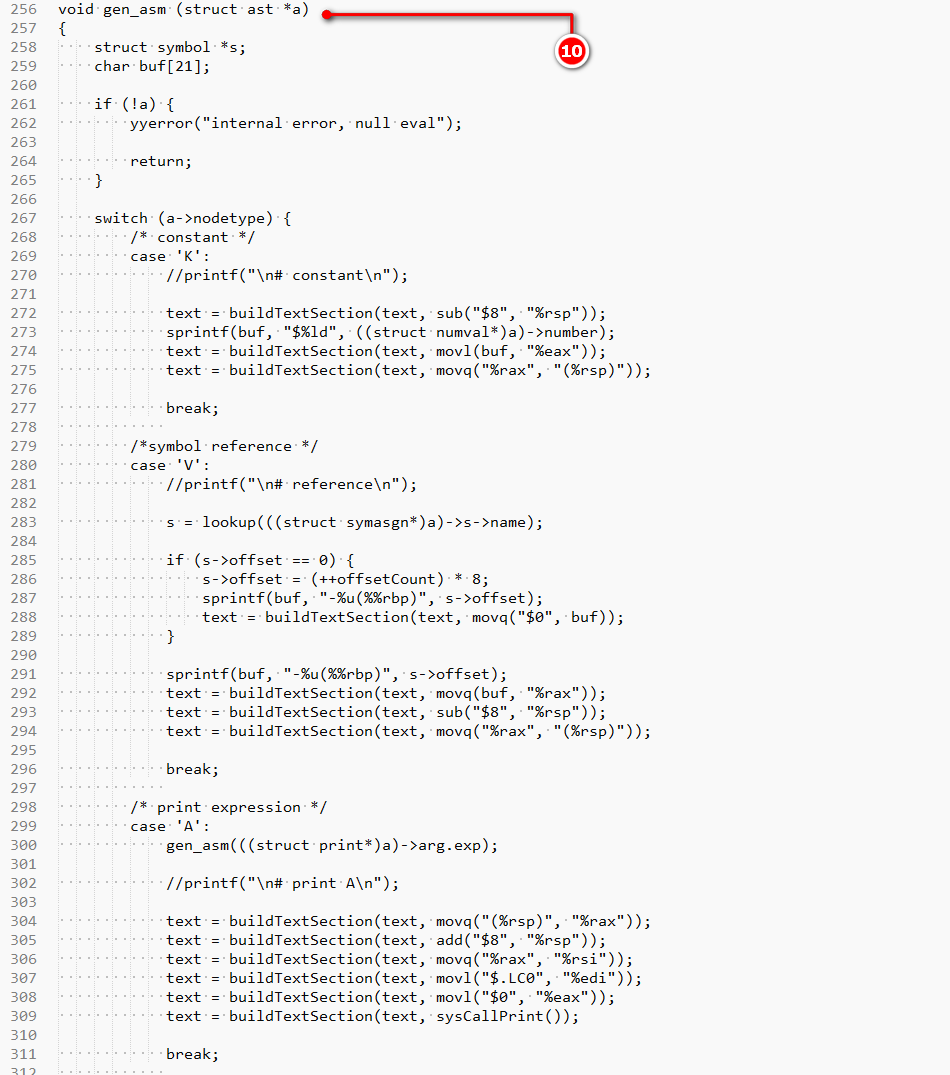
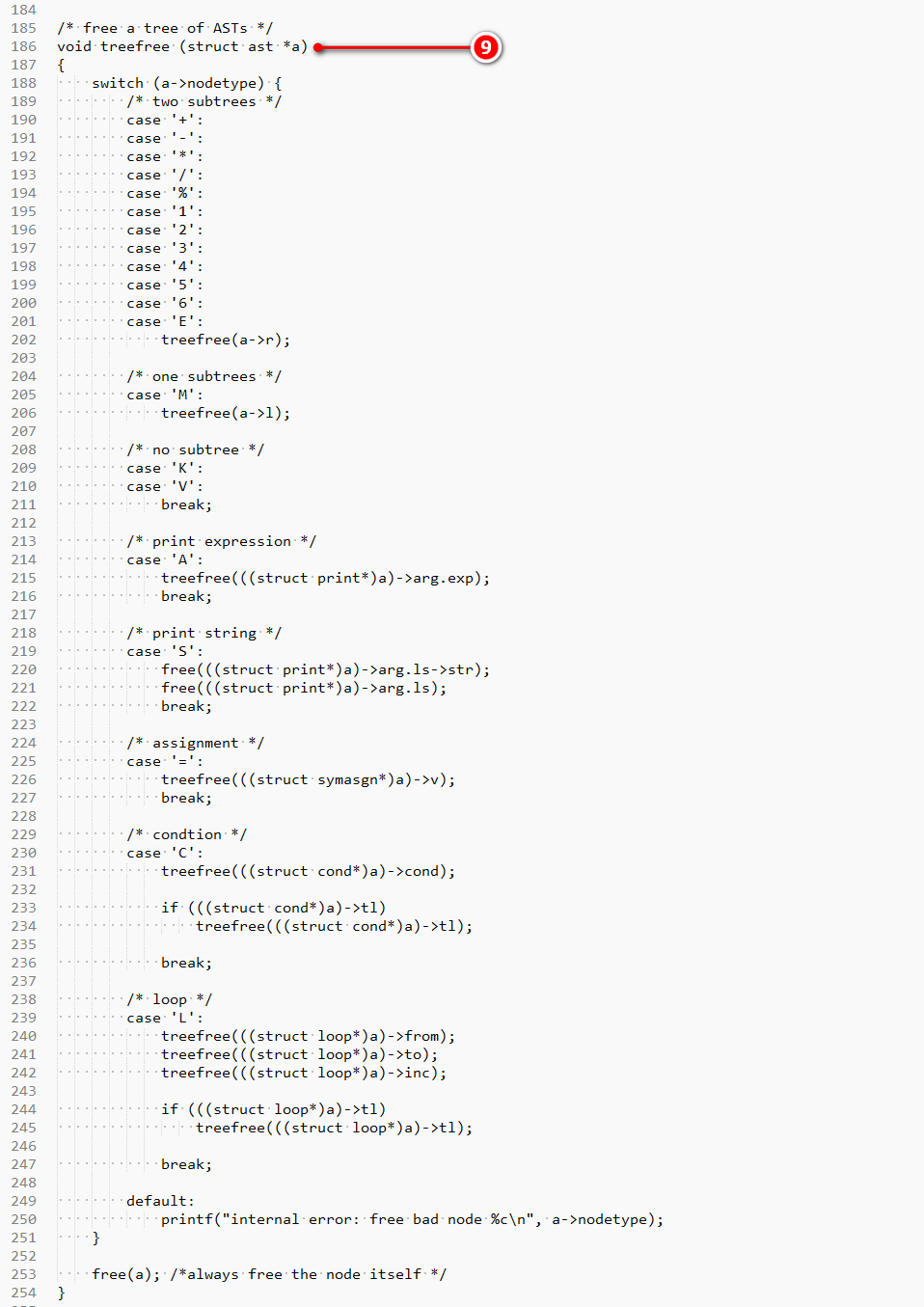
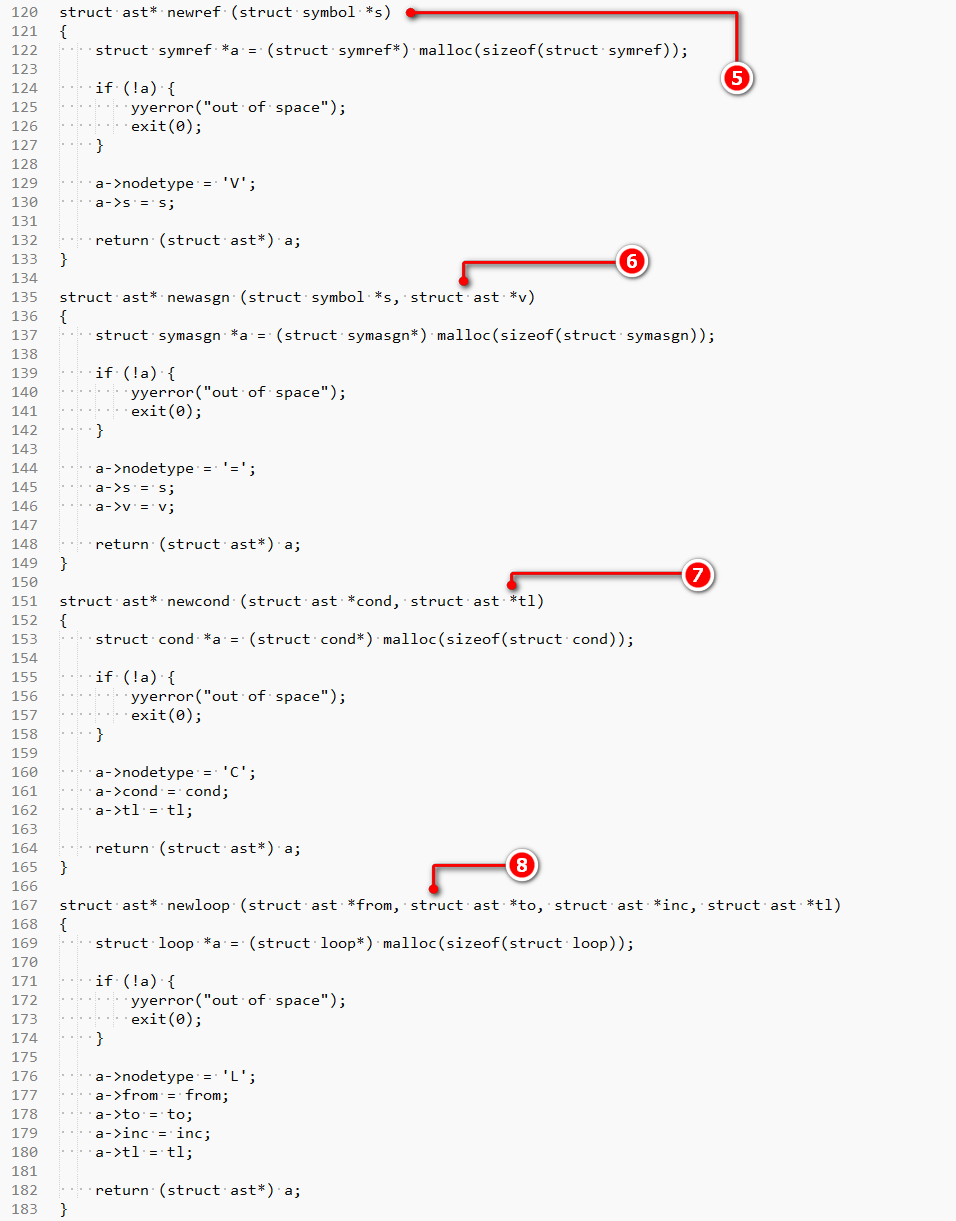
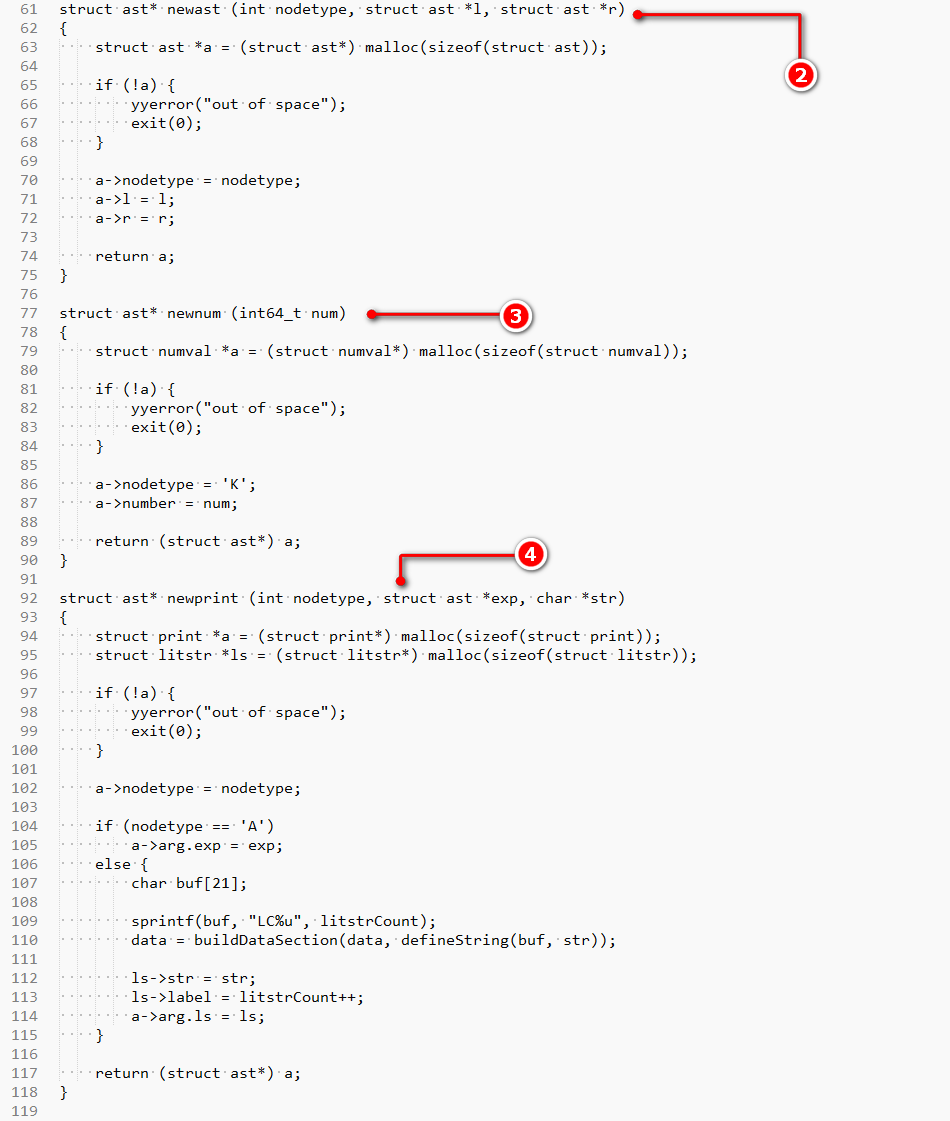
**Ast prototype File : elecompiler.h**



**อธิบายหลักการและส่วนหลักๆของไฟล์ elecompiler.h**

* (1) Struct ของ symbol table
* (2) Struct ของ Abstract syntax tree
* (3,4,5,6,7,8,9) Struct ของ type ของ node แต่ละชนิด ที่จะต้องมี Struct ของ Abstract syntax tree เพื่อที่จะให้สามารถต่อ tree ไปแต่ละ Node ได้ ซึ่ง Attribute ของแต่ละ Node จะแตกต่างกันไปตาม แต่ละชนิดข้อมูลที่ได้จาก การทำ Parser

**Main Compiler File : elecompiler.c**



**อธิบายหลักการและส่วนหลักๆของไฟล์ elecompiler.c**

* (1) struct symbol\* lookup (char \*sym)

ใช้ในการ lookup symbol table ที่ implement แบบ hash table ถ้าไม่เจอจะทำการ add symbol ใหม่เข้าไป

* (2) struct ast\* newast (int nodetype, struct ast \*l, struct ast \*r)

สร้าง node ใน AST ตัวใหม่ขึ้นมา โดยมี child สองตัวคือ left (l) และ right (r)

* (3) struct ast\* newnum (int64\_t num)

สร้าง node ใน AST ตัวใหม่ขึ้นมา ใช้เก็บค่าคงที่ (literal)

* (4) struct ast\* newprint (int nodetype, struct ast \*exp, char \*str)

สร้าง node ใน AST ตัวใหม่ขึ้นมา ใช้ในการเก็บ print() statement โดย node ชนิดนี้จะมีอยู่สองแบบ คือ 1. ใช้สำหรับ print literal string และ 2. ใช้สำหรับ print ค่าตัวแปร

* (5) struct ast\* newref (struct symbol \*s)

สร้าง node ใน AST ตัวใหม่ขึ้นมา ใช้ในการอ้างถึงตัวแปร (symbol)

* (6) struct ast\* newasgn (struct symbol \*s, struct ast \*v)

สร้าง node ใน AST ตัวใหม่ขึ้นมา ใช้เก็บ assignment statement symbol (s) และ expression (v)

* (7) struct ast\* newcond (struct ast \*cond, struct ast \*tl)

สร้าง node ใน AST ตัวใหม่ขึ้นมา ใช้เก็บ if-then statement โดยมี child เป็น condition และ then (tl)

* (8) struct ast\* newloop (struct ast \*from, struct ast \*to, struct ast \*inc, struct ast \*tl)

สร้าง node ใน AST ตัวใหม่ขึ้นมา ใช้เก็บ loop statement โดยมี child สามตัว คือ 1. เลขตั้งต้น (from)

2. เลขสุดท้าย (to) และ 3. จำนวนที่จะใช้บวกเข้าไปกับเลขตั้งต้น (increment)

* (9) void treefree (struct ast \*a)

ใช้ในการ free AST ที่ได้สร้างขึ้นมาในขั้นตอน parsing

* (10) void gen\_asm (struct ast \*a)

ใช้ในการ walk through AST และสร้าง file assembly

* (11) int64\_t eval (struct ast \*a)

ใช้ evaluate tree ในที่นี้หมายถึง AST

* (12) int main(int argc, char \*\*argv)  
  Function หลักที่ใช้ในการทำงานของ program

**4.ผลการรัน**

**4.1 ผลการันกับตัวอย่างที่ผิด**

1. 1 + “test”

Error เนื่องจากไม่สามารถ + ค่า int กับ string ได้

2. loop ( 1->10 , a ){ I = 1; }

Error เนื่องจากตำแหน่งของ a ใน gramma เป็น Constant ไม่ใช่ Variable

3. “test” = 3

Error เนื่องจาก ไม่สามารถ assign ค่าให้ variable string ได้

4. If ( “x” == “x”)

Error เนื่องจาก ไม่สามารถ check condition ระหว่าง string ได้

5. Print if(x =8)

Error เนื่องจาก ไม่สามารถ ptint statement ได้

**4.2 ผลการันกับตัวอย่างที่ถูก**

**1.** a = (1+2)\*2;

**Assembly Code**

7 from:

8 .zero 8

9 .section .rodata

10 .LC0:

11 .string "%ld"

12 .LC1:

13 .string "%s"

14 .text

15 .globl main

16 .type main, @function

17 main:

18 .LFB0:

19 .cfi\_startproc

20 pushq %rbp

21 .cfi\_def\_cfa\_offset 16

22 .cfi\_offset 6, -16

23 movq %rsp, %rbp

24 .cfi\_def\_cfa\_register 6

25 subq $16, %rsp

26 subq $8, %rsp

27 movl $2, %eax

28 movq %rax, (%rsp)

29 subq $8, %rsp

30 movl $2, %eax

31 movq %rax, (%rsp)

32 subq $8, %rsp

33 movl $1, %eax

34 movq %rax, (%rsp)

35 movq (%rsp), %rdx

36 addq $8, %rsp

37 movq (%rsp), %rax

38 addq $8, %rsp

39 addq %rdx, %rax

40 subq $8, %rsp

41 movq %rax, (%rsp)

42 movq (%rsp), %rax

43 addq $8, %rsp

44 imulq (%rsp), %rax

45 movq %rax, (%rsp)

46 movq (%rsp), %rax

47 addq $8, %rsp

48 movq %rax, -8(%rbp)

49 subq $8, %rsp

50 movq %rax, (%rsp)

51 movl $0, %eax

52 leave

53 .cfi\_def\_cfa 7, 8

54 ret

55 .cfi\_endproc

56 .LFE0:

57 .size main, .-main

58 .ident "GCC: (GNU) 6.1.1 20160501"

59 .section .note.GNU-stack,"",@progbits

**2.** a = 1\*2+3;  
 a = a\*a;

**Assembly Code**

1 .file "test.c"

2 .globl from

3 .bss

4 .align 8

5 .type from, @object

6 .size from, 8

7 from:

8 .zero 8

9 .section .rodata

10 .LC0:

11 .string "%ld"

12 .LC1:

13 .string "%s"

14 .text

15 .globl main

16 .type main, @function

17 main:

18 .LFB0:

19 .cfi\_startproc

20 pushq %rbp

21 .cfi\_def\_cfa\_offset 16

22 .cfi\_offset 6, -16

23 movq %rsp, %rbp

24 .cfi\_def\_cfa\_register 6

25 subq $16, %rsp

26 subq $8, %rsp

27 movl $3, %eax

28 movq %rax, (%rsp)

29 subq $8, %rsp

30 movl $2, %eax

31 movq %rax, (%rsp)

32 subq $8, %rsp

33 movl $1, %eax

34 movq %rax, (%rsp)

35 movq (%rsp), %rax

36 addq $8, %rsp

37 imulq (%rsp), %rax

38 movq %rax, (%rsp)

39 movq (%rsp), %rdx

40 addq $8, %rsp

41 movq (%rsp), %rax

42 addq $8, %rsp

43 addq %rdx, %rax

44 subq $8, %rsp

45 movq %rax, (%rsp)

46 movq (%rsp), %rax

47 addq $8, %rsp

48 movq %rax, -8(%rbp)

49 subq $8, %rsp

50 movq %rax, (%rsp)

51 movq -8(%rbp), %rax

52 subq $8, %rsp

53 movq %rax, (%rsp)

54 movq -8(%rbp), %rax

55 subq $8, %rsp

56 movq %rax, (%rsp)

57 movq (%rsp), %rax

58 addq $8, %rsp

59 imulq (%rsp), %rax

60 movq %rax, (%rsp)

61 movq (%rsp), %rax

62 addq $8, %rsp

63 movq %rax, -8(%rbp)

64 subq $8, %rsp

65 movq %rax, (%rsp)

66 movl $0, %eax

67 leave

68 .cfi\_def\_cfa 7, 8

69 ret

70 .cfi\_endproc

71 .LFE0:

72 .size main, .-main

73 .ident "GCC: (GNU) 6.1.1 20160501"

74 .section .note.GNU-stack,"",@progbits

**3.** a = (1\*2\*3)%4;  
 print (a\*3);

**Assembly Code**

1 .file "test.c"

2 .globl from

3 .bss

4 .align 8

5 .type from, @object

6 .size from, 8

7 from:

8 .zero 8

9 .section .rodata

10 .LC0:

11 .string "%ld"

12 .LC1:

13 .string "%s"

14 .text

15 .globl main

16 .type main, @function

17 main:

18 .LFB0:

19 .cfi\_startproc

20 pushq %rbp

21 .cfi\_def\_cfa\_offset 16

22 .cfi\_offset 6, -16

23 movq %rsp, %rbp

24 .cfi\_def\_cfa\_register 6

25 subq $16, %rsp

26 subq $8, %rsp

27 movl $4, %eax

28 movq %rax, (%rsp)

29 subq $8, %rsp

30 movl $3, %eax

31 movq %rax, (%rsp)

32 subq $8, %rsp

33 movl $2, %eax

34 movq %rax, (%rsp)

35 subq $8, %rsp

36 movl $1, %eax

37 movq %rax, (%rsp)

38 movq (%rsp), %rax

39 addq $8, %rsp

40 imulq (%rsp), %rax

41 movq %rax, (%rsp)

42 movq (%rsp), %rax

43 addq $8, %rsp

44 imulq (%rsp), %rax

45 movq %rax, (%rsp)

46 movq (%rsp), %rax

47 addq $8, %rsp

48 cqto

49 idivq (%rsp)

50 movq %rdx, (%rsp)

51 movq (%rsp), %rax

52 addq $8, %rsp

53 movq %rax, -8(%rbp)

54 subq $8, %rsp

55 movq %rax, (%rsp)

56 subq $8, %rsp

57 movl $3, %eax

58 movq %rax, (%rsp)

59 movq -8(%rbp), %rax

60 subq $8, %rsp

61 movq %rax, (%rsp)

62 movq (%rsp), %rax

63 addq $8, %rsp

64 imulq (%rsp), %rax

65 movq %rax, (%rsp)

66 movq (%rsp), %rax

67 addq $8, %rsp

68 movq %rax, %rsi

69 movl $.LC0, %edi

70 movl $0, %eax

71 call printf

72 movl $0, %eax

73 leave

74 .cfi\_def\_cfa 7, 8

75 ret

76 .cfi\_endproc

77 .LFE0:

78 .size main, .-main

79 .ident "GCC: (GNU) 6.1.1 20160501"

80 .section .note.GNU-stack,"",@progbits

**Output**



**4.** a = -4;  
 if(-a){  
 print (a);\  
 }  
 print ("a >v 0");

**Assembly Code**

1 .file "test.c"

2 .globl from

3 .bss

4 .align 8

5 .type from, @object

6 .size from, 8

7 from:

8 .zero 8

9 .section .rodata

10 .LC0:

11 .string "%ld"

12 .LC1:

13 .string "%s"

14 .LC2:

15 .string "a >v 0"

16 .text

17 .globl main

18 .type main, @function

19 main:

20 .LFB0:

21 .cfi\_startproc

22 pushq %rbp

23 .cfi\_def\_cfa\_offset 16

24 .cfi\_offset 6, -16

25 movq %rsp, %rbp

26 .cfi\_def\_cfa\_register 6

27 subq $16, %rsp

28 subq $8, %rsp

29 movl $4, %eax

30 movq %rax, (%rsp)

31 movq (%rsp), %rax

32 negq %rax

33 movq %rax, (%rsp)

34 movq (%rsp), %rax

35 addq $8, %rsp

36 movq %rax, -8(%rbp)

37 subq $8, %rsp

38 movq %rax, (%rsp)

39 movq -8(%rbp), %rax

40 subq $8, %rsp

41 movq %rax, (%rsp)

42 movq (%rsp), %rax

43 negq %rax

44 movq %rax, (%rsp)

45 cmpq $0, (%rsp)

46 je .L0

47 movq -8(%rbp), %rax

48 subq $8, %rsp

49 movq %rax, (%rsp)

50 movq (%rsp), %rax

51 addq $8, %rsp

52 movq %rax, %rsi

53 movl $.LC0, %edi

54 movl $0, %eax

55 call printf

56 .L0:

57 movl $.LC2, %esi

58 movl $.LC1, %edi

59 movl $0, %eax

60 call printf

61 movl $0, %eax

62 leave

63 .cfi\_def\_cfa 7, 8

64 ret

65 .cfi\_endproc

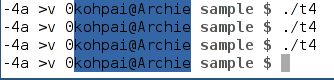
66 .LFE0:

67 .size main, .-main

68 .ident "GCC: (GNU) 6.1.1 20160501"

69 .section .note.GNU-stack,"",@progbits

**Output**



**5.** a = -100000;  
 if(a){  
 print ("a < 0");\  
 }  
 if(-a){

}  
 print ("a > 0");

**Assembly Code**

1 .file "test.c"

2 .globl from

3 .bss

4 .align 8

5 .type from, @object

6 .size from, 8

7 from:

8 .zero 8

9 .section .rodata

10 .LC0:

11 .string "%ld"

12 .LC1:

13 .string "%s"

14 .LC2:

15 .string "a < 0"

16 .LC3:

17 .string "a > 0"

18 .text

19 .globl main

20 .type main, @function

21 main:

22 .LFB0:

23 .cfi\_startproc

24 pushq %rbp

25 .cfi\_def\_cfa\_offset 16

26 .cfi\_offset 6, -16

27 movq %rsp, %rbp

28 .cfi\_def\_cfa\_register 6

29 subq $16, %rsp

30 subq $8, %rsp

31 movl $100000, %eax

32 movq %rax, (%rsp)

33 movq (%rsp), %rax

34 negq %rax

35 movq %rax, (%rsp)

36 movq (%rsp), %rax

37 addq $8, %rsp

38 movq %rax, -8(%rbp)

39 subq $8, %rsp

40 movq %rax, (%rsp)

41 movq -8(%rbp), %rax

42 subq $8, %rsp

43 movq %rax, (%rsp)

44 cmpq $0, (%rsp)

45 je .L0

46 movl $.LC2, %esi

47 movl $.LC1, %edi

48 movl $0, %eax

49 call printf

50 .L0:

51 movq -8(%rbp), %rax

52 subq $8, %rsp

53 movq %rax, (%rsp)

54 movq (%rsp), %rax

55 negq %rax

56 movq %rax, (%rsp)

57 cmpq $0, (%rsp)

58 je .L1

59 .L1:

60 movl $.LC3, %esi

61 movl $.LC1, %edi

62 movl $0, %eax

63 call printf

64 movl $0, %eax

65 leave

66 .cfi\_def\_cfa 7, 8

67 ret

68 .cfi\_endproc

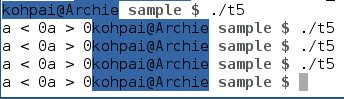
69 .LFE0:

70 .size main, .-main

71 .ident "GCC: (GNU) 6.1.1 20160501"

72 .section .note.GNU-stack,"",@progbits

**Output**



**6. i = 1;  
 loop (0->10,1){  
 I = i+1;\  
 print (i\*3);\  
 }**

**Assembly Code**

1 .file "test.c"

2 .globl from

3 .bss

4 .align 8

5 .type from, @object

6 .size from, 8

7 from:

8 .zero 8

9 .section .rodata

10 .LC0:

11 .string "%ld"

12 .LC1:

13 .string "%s"

14 .text

15 .globl main

16 .type main, @function

17 main:

18 .LFB0:

19 .cfi\_startproc

20 pushq %rbp

21 .cfi\_def\_cfa\_offset 16

22 .cfi\_offset 6, -16

23 movq %rsp, %rbp

24 .cfi\_def\_cfa\_register 6

25 subq $16, %rsp

26 subq $8, %rsp

27 movl $1, %eax

28 movq %rax, (%rsp)

29 movq (%rsp), %rax

30 addq $8, %rsp

31 movq %rax, -8(%rbp)

32 subq $8, %rsp

33 movq %rax, (%rsp)

34 movl $0, %eax

35 movq %rax, from(%rip)

36 jmp .L0

37 .L1:

38 subq $8, %rsp

39 movl $1, %eax

40 movq %rax, (%rsp)

41 movq -8(%rbp), %rax

42 subq $8, %rsp

43 movq %rax, (%rsp)

44 movq (%rsp), %rdx

45 addq $8, %rsp

46 movq (%rsp), %rax

47 addq $8, %rsp

48 addq %rdx, %rax

49 subq $8, %rsp

50 movq %rax, (%rsp)

51 movq (%rsp), %rax

52 addq $8, %rsp

53 movq %rax, -8(%rbp)

54 subq $8, %rsp

55 movq %rax, (%rsp)

56 subq $8, %rsp

57 movl $3, %eax

58 movq %rax, (%rsp)

59 movq -8(%rbp), %rax

60 subq $8, %rsp

61 movq %rax, (%rsp)

62 movq (%rsp), %rax

63 addq $8, %rsp

64 imulq (%rsp), %rax

65 movq %rax, (%rsp)

66 movq (%rsp), %rax

67 addq $8, %rsp

68 movq %rax, %rsi

69 movl $.LC0, %edi

70 movl $0, %eax

71 call printf

72 movq from(%rip), %rdx

73 movl $1, %eax

74 addq %rdx, %rax

75 movq %rax, from(%rip)

76 .L0:

77 movl $10, %eax

78 cmpq %rax, from(%rip)

79 jle .L1

80 movl $0, %eax

81 leave

82 .cfi\_def\_cfa 7, 8

83 ret

84 .cfi\_endproc

85 .LFE0:

86 .size main, .-main

87 .ident "GCC: (GNU) 6.1.1 20160501"

88 .section .note.GNU-stack,"",@progbits

**Output**



**7**. i = 1;  
 j = 0;  
 loop (0->10,1){  
 i = i+1;\  
 j=i\*i;\  
 print (i\*3);\  
 }  
 if(i-100){  
 print ("a > 100");\  
 }

**Assembly Code**

1 .file "test.c"

2 .globl from

3 .bss

4 .align 8

5 .type from, @object

6 .size from, 8

7 from:

8 .zero 8

9 .section .rodata

10 .LC0:

11 .string "%ld"

12 .LC1:

13 .string "%s"

14 .LC2:

15 .string "a > 100"

16 .text

17 .globl main

18 .type main, @function

19 main:

20 .LFB0:

21 .cfi\_startproc

22 pushq %rbp

23 .cfi\_def\_cfa\_offset 16

24 .cfi\_offset 6, -16

25 movq %rsp, %rbp

26 .cfi\_def\_cfa\_register 6

27 subq $24, %rsp

28 subq $8, %rsp

29 movl $1, %eax

30 movq %rax, (%rsp)

31 movq (%rsp), %rax

32 addq $8, %rsp

33 movq %rax, -8(%rbp)

34 subq $8, %rsp

35 movq %rax, (%rsp)

36 subq $8, %rsp

37 movl $0, %eax

38 movq %rax, (%rsp)

39 movq (%rsp), %rax

40 addq $8, %rsp

41 movq %rax, -16(%rbp)

42 subq $8, %rsp

43 movq %rax, (%rsp)

44 movl $0, %eax

45 movq %rax, from(%rip)

46 jmp .L0

47 .L1:

48 subq $8, %rsp

49 movl $1, %eax

50 movq %rax, (%rsp)

51 movq -8(%rbp), %rax

52 subq $8, %rsp

53 movq %rax, (%rsp)

54 movq (%rsp), %rdx

55 addq $8, %rsp

56 movq (%rsp), %rax

57 addq $8, %rsp

58 addq %rdx, %rax

59 subq $8, %rsp

60 movq %rax, (%rsp)

61 movq (%rsp), %rax

62 addq $8, %rsp

63 movq %rax, -8(%rbp)

64 subq $8, %rsp

65 movq %rax, (%rsp)

66 movq -8(%rbp), %rax

67 subq $8, %rsp

68 movq %rax, (%rsp)

69 movq -8(%rbp), %rax

70 subq $8, %rsp

71 movq %rax, (%rsp)

72 movq (%rsp), %rax

73 addq $8, %rsp

74 imulq (%rsp), %rax

75 movq %rax, (%rsp)

76 movq (%rsp), %rax

77 addq $8, %rsp

78 movq %rax, -16(%rbp)

79 subq $8, %rsp

80 movq %rax, (%rsp)

81 subq $8, %rsp

82 movl $3, %eax

83 movq %rax, (%rsp)

84 movq -8(%rbp), %rax

85 subq $8, %rsp

86 movq %rax, (%rsp)

87 movq (%rsp), %rax

88 addq $8, %rsp

89 imulq (%rsp), %rax

90 movq %rax, (%rsp)

91 movq (%rsp), %rax

92 addq $8, %rsp

93 movq %rax, %rsi

94 movl $.LC0, %edi

95 movl $0, %eax

96 call printf

97 movq from(%rip), %rdx

98 movl $1, %eax

99 addq %rdx, %rax

100 movq %rax, from(%rip)

101 .L0:

102 movl $10, %eax

103 cmpq %rax, from(%rip)

104 jle .L1

105 subq $8, %rsp

106 movl $100, %eax

107 movq %rax, (%rsp)

108 movq -8(%rbp), %rax

109 subq $8, %rsp

110 movq %rax, (%rsp)

111 movq (%rsp), %rax

112 addq $8, %rsp

113 subq (%rsp), %rax

114 movq %rax, (%rsp)

115 cmpq $0, (%rsp)

116 je .L2

117 movl $.LC2, %esi

118 movl $.LC1, %edi

119 movl $0, %eax

120 call printf

121 .L2:

122 movl $0, %eax

123 leave

124 .cfi\_def\_cfa 7, 8

125 ret

126 .cfi\_endproc

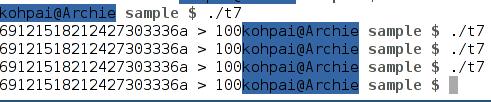
127 .LFE0:

128 .size main, .-main

129 .ident "GCC: (GNU) 6.1.1 20160501"

130 .section .note.GNU-stack,"",@progbits

**Output**



**8.** **i = 1;** loop (0->10,1){\  
 i = i+1;\  
 print (i\*3);\  
 print("\n");\  
 }

**Assembly Code**

1 .file "test.c"

2 .globl from

3 .bss

4 .align 8

5 .type from, @object

6 .size from, 8

7 from:

8 .zero 8

9 .section .rodata

10 .LC0:

11 .string "%ld"

12 .LC1:

13 .string "%s"

14 .LC2:

15 .string "\n"

16 .text

17 .globl main

18 .type main, @function

19 main:

20 .LFB0:

21 .cfi\_startproc

22 pushq %rbp

23 .cfi\_def\_cfa\_offset 16

24 .cfi\_offset 6, -16

25 movq %rsp, %rbp

26 .cfi\_def\_cfa\_register 6

27 subq $16, %rsp

28 subq $8, %rsp

29 movl $1, %eax

30 movq %rax, (%rsp)

31 movq (%rsp), %rax

32 addq $8, %rsp

33 movq %rax, -8(%rbp)

34 subq $8, %rsp

35 movq %rax, (%rsp)

36 movl $0, %eax

37 movq %rax, from(%rip)

38 jmp .L0

39 .L1:

40 subq $8, %rsp

41 movl $1, %eax

42 movq %rax, (%rsp)

43 movq -8(%rbp), %rax

44 subq $8, %rsp

45 movq %rax, (%rsp)

46 movq (%rsp), %rdx

47 addq $8, %rsp

48 movq (%rsp), %rax

49 addq $8, %rsp

50 addq %rdx, %rax

51 subq $8, %rsp

52 movq %rax, (%rsp)

53 movq (%rsp), %rax

54 addq $8, %rsp

55 movq %rax, -8(%rbp)

56 subq $8, %rsp

57 movq %rax, (%rsp)

58 movl $.LC2, %esi

59 movl $.LC1, %edi

60 movl $0, %eax

61 call printf

62 movq from(%rip), %rdx

63 movl $1, %eax

64 addq %rdx, %rax

65 movq %rax, from(%rip)

66 .L0:

67 movl $10, %eax

68 cmpq %rax, from(%rip)

69 jle .L1

70 movl $0, %eax

71 leave

72 .cfi\_def\_cfa 7, 8

73 ret

74 .cfi\_endproc

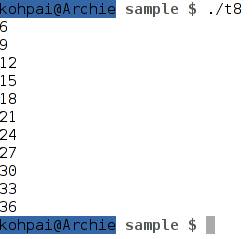
75 .LFE0:

76 .size main, .-main

77 .ident "GCC: (GNU) 6.1.1 20160501"

78 .section .note.GNU-stack,"",@progbits

**Output**



**9.** **i = 1;** j = i+1;  
 print ("i+j : ");  
 print(i+j);  
 print("\n");

**Assembly Code**

1 .file "test.c"

2 .globl from

3 .bss

4 .align 8

5 .type from, @object

6 .size from, 8

7 from:

8 .zero 8

9 .section .rodata

10 .LC0:

11 .string "%ld"

12 .LC1:

13 .string "%s"

14 .LC2:

15 .string "i+j : "

16 .LC3:

17 .string "\n"

18 .text

19 .globl main

20 .type main, @function

21 main:

22 .LFB0:

23 .cfi\_startproc

24 pushq %rbp

25 .cfi\_def\_cfa\_offset 16

26 .cfi\_offset 6, -16

27 movq %rsp, %rbp

28 .cfi\_def\_cfa\_register 6

29 subq $24, %rsp

30 subq $8, %rsp

31 movl $1, %eax

32 movq %rax, (%rsp)

33 movq (%rsp), %rax

34 addq $8, %rsp

35 movq %rax, -8(%rbp)

36 subq $8, %rsp

37 movq %rax, (%rsp)

38 subq $8, %rsp

39 movl $1, %eax

40 movq %rax, (%rsp)

41 movq -8(%rbp), %rax

42 subq $8, %rsp

43 movq %rax, (%rsp)

44 movq (%rsp), %rdx

45 addq $8, %rsp

46 movq (%rsp), %rax

47 addq $8, %rsp

48 addq %rdx, %rax

49 subq $8, %rsp

50 movq %rax, (%rsp)

51 movq (%rsp), %rax

52 addq $8, %rsp

53 movq %rax, -16(%rbp)

54 subq $8, %rsp

55 movq %rax, (%rsp)

56 movl $.LC2, %esi

57 movl $.LC1, %edi

58 movl $0, %eax

59 call printf

60 movq -16(%rbp), %rax

61 subq $8, %rsp

62 movq %rax, (%rsp)

63 movq -8(%rbp), %rax

64 subq $8, %rsp

65 movq %rax, (%rsp)

66 movq (%rsp), %rdx

67 addq $8, %rsp

68 movq (%rsp), %rax

69 addq $8, %rsp

70 addq %rdx, %rax

71 subq $8, %rsp

72 movq %rax, (%rsp)

73 movq (%rsp), %rax

74 addq $8, %rsp

75 movq %rax, %rsi

76 movl $.LC0, %edi

77 movl $0, %eax

78 call printf

79 movl $.LC3, %esi

80 movl $.LC1, %edi

81 movl $0, %eax

82 call printf

83 movl $0, %eax

84 leave

85 .cfi\_def\_cfa 7, 8

86 ret

87 .cfi\_endproc

88 .LFE0:

89 .size main, .-main

90 .ident "GCC: (GNU) 6.1.1 20160501"

91 .section .note.GNU-stack,"",@progbits

**Output**



**10.** i = 1;  
 print (i);  
 i = 10;  
 print (i);

**Assembly Code**

1 .file "test.c"

2 .globl from

3 .bss

4 .align 8

5 .type from, @object

6 .size from, 8

7 from:

8 .zero 8

9 .section .rodata

10 .LC0:

11 .string "%ld"

12 .LC1:

13 .string "%s"

14 .text

15 .globl main

16 .type main, @function

17 main:

18 .LFB0:

19 .cfi\_startproc

20 pushq %rbp

21 .cfi\_def\_cfa\_offset 16

22 .cfi\_offset 6, -16

23 movq %rsp, %rbp

24 .cfi\_def\_cfa\_register 6

25 subq $16, %rsp

26 subq $8, %rsp

27 movl $1, %eax

28 movq %rax, (%rsp)

29 movq (%rsp), %rax

30 addq $8, %rsp

31 movq %rax, -8(%rbp)

32 subq $8, %rsp

33 movq %rax, (%rsp)

34 movq -8(%rbp), %rax

35 subq $8, %rsp

36 movq %rax, (%rsp)

37 movq (%rsp), %rax

38 addq $8, %rsp

39 movq %rax, %rsi

40 movl $.LC0, %edi

41 movl $0, %eax

42 call printf

43 subq $8, %rsp

44 movl $10, %eax

45 movq %rax, (%rsp)

46 movq (%rsp), %rax

47 addq $8, %rsp

48 movq %rax, -8(%rbp)

49 subq $8, %rsp

50 movq %rax, (%rsp)

51 movq -8(%rbp), %rax

52 subq $8, %rsp

53 movq %rax, (%rsp)

54 movq (%rsp), %rax

55 addq $8, %rsp

56 movq %rax, %rsi

57 movl $.LC0, %edi

58 movl $0, %eax

59 call printf

60 movl $0, %eax

61 leave

62 .cfi\_def\_cfa 7, 8

63 ret

64 .cfi\_endproc

65 .LFE0:

66 .size main, .-main

67 .ident "GCC: (GNU) 6.1.1 20160501"

68 .section .note.GNU-stack,"",@progbits

**Output**

