/*lexfile*/

```
D                       [0-9]
L                       [a-zA-Z_]
H                       [a-fA-F0-9]
E                       [Ee][+-]?{D}+
FS                      (f|F|l|L)
IS                      (u|U|l|L)*

%{
#include <stdio.h>
#include "mylang.tab.h"
extern FILE * fp;
void count();
%}

%%

"break"                 { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","BREAK"); return(BREAK);}
"char"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","CHAR"); return(CHAR);}
"continue"              { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","CONTINUE"); return(CONTINUE);}
"else"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","ELSE"); return(ELSE);}
"float"                 { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","FLOAT"); return(FLOAT);}
"for"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","FOR"); return(FOR);}
"if"                    { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","IF"); return(IF);}
"int"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","INT"); return(INT);}
"return"                { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","RETURN"); return(RETURN);}
"void"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","VOID"); return(VOID);}
"while"                 { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","WHILE"); return(WHILE);}
"proc"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","PROC"); return(PROC);}
"lnk"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","LNK"); return(LNK);}
"jb"                    { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","JB"); return(JB);}
"clust"                 { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","CLUST"); return(CLUST);}
"Cluster"               { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","CLUSTER"); return(CLUSTER);}
"Processor"             { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","PROCESSOR"); return(PROCESSOR);}
"processors"    { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","PROCESSORS"); return(PROCESSORS);}
"isa"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","ISA"); return(ISA);}
('ARM')|('AMD')|('CDC')|('MIPS') { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","PROC_TYPE"); return(PROC_TYPE);}
"clock_speed" { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","CLOCK_SPEED"); return(CLOCK_SPEED);}
"l1_memory"             { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","MEM1"); return(MEM1);}
"l2_memory"             { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","MEM2"); return(MEM2);}
"name"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","NAME"); return(NAME);}
"topology"              { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","TOPOLOGY"); return(TOPOLOGY);}
```

```
"link_bandwidth"  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","LINK_BANDWIDTH"); return(LINK_BANDWIDTH);}
"link_capacity"{ fprintf(fp,"<"); count(); fprintf(fp,",%s> ","LINK_CAPACITY"); return(LINK_CAPACITY);}
"Link"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","LINK"); return(LINK);}
"start_point"    { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","START_POINT"); return(START_POINT);}
"end_point"             { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","END_POINT"); return(END_POINT);}
"memory_type"           { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","MEMORY_TYPE"); return(MEMORY_TYPE);}
('primary')|('secondary')|('cache') { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","MEM_TYPES"); return(MEM_TYPES);}
"mem_size"              { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","MEMORY_SIZE"); return(MEMORY_SIZE);}
"Job"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","JOB"); return(JOB);}
"job_id"                { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","JOB_ID"); return(JOB_ID);}
"flops_required" { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","FLOPS_REQUIRED"); return(FLOPS_REQUIRED);}
"deadline"              { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","DEADLINE"); return(DEADLINE);}
"mem_required"          { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","MEM_REQUIRED"); return(MEM_REQUIRED);}
"affinity"              { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","AFFINITY"); return(AFFINITY);}
"run"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","RUN"); return(RUN);}
"wait"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","WAIT"); return(WAIT);}
"discard_job"   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","DISCARD_JOB"); return(DISCARD_JOB);}
"stop"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","STOP"); return(STOP);}
"get_available_memory" { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","GET_AVAILABLE_MEMORY"); return(GET_AVAILABLE_MEMORY);}
"get_job_affinity"              { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","GET_JOB_AFFINITY"); return(GET_JOB_AFFINITY);}
"get_memory"            { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","GET_JOB_MEMORY"); return(GET_JOB_MEMORY);}
"get_flops"                             { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","GET_FLOPS"); return(GET_FLOPS);}
"get_deadline"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","GET_DEADLINE"); return(GET_DEADLINE);}
"is_running"                    { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","IS_RUNNING"); return(IS_RUNNING);}
"submit_jobs"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","SUBMIT_JOBS"); return(SUBMIT_JOBS);}
"get_flops_speed"               { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","GET_FLOPS_SPEED"); return(GET_FLOPS_SPEED);}
"get_proc_type"                         { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","GET_PROC_TYPE"); return(GET_PROC_TYPE);}
"is_processor"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","IS_PROCESSOR"); return(IS_PROCESSOR);}
"get_processor"                         { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","GET_PROCESSOR"); return(GET_PROCESSOR);}
"Memory"                                { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","MEMORY"); return(MEMORY);}
"get_clock_speed" { fprintf(fp,"<"); count(); fprintf(fp,",%s ","GET_CLOCK_SPEEDD"); return(GET_CLOCK_SPEED);}
"mem"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","MEM"); return(MEM);}

{L}({L}|{D})*           { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","IDENTIFIER"); return(IDENTIFIER);}

0[xX]{H}+{IS}?          { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","CONSTANT"); return(CONSTANT);}
0{D}+{IS}?              { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","CONSTANT"); return(CONSTANT);}
{D}+{IS}?               { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","CONSTANT"); return(CONSTANT);}
L?'(\\.|[^\\'])+'   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","CONSTANT"); return(CONSTANT);}

{D}+{E}{FS}?            { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","CONSTANT"); return(CONSTANT);}
{D}*"."{D}+({E})?{FS}?{ fprintf(fp,"<"); count(); fprintf(fp,",%s> ","CONSTANT"); return(CONSTANT);}
{D}+"."{D}*({E})?{FS}?{ fprintf(fp,"<"); count(); fprintf(fp,",%s> ","CONSTANT"); return(CONSTANT);}
```

```
\"(\\.|[^\\"])*\"     { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","STRING_LITERAL"); return(STRING_LITERAL);}

">>"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","RIGHT_OP"); return(RIGHT_OP);}
"<<"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","LEFT_OP"); return(LEFT_OP);}
"++"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","INC_OP"); return(INC_OP);}
"--"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","DEC_OP"); return(DEC_OP);}
"->"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","DEREF_OP"); return(DEREF_OP);}
"&&"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","AND_OP"); return(AND_OP);}
"||"                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","OR_OP"); return(OR_OP);}
"<="                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","LE_OP"); return(LE_OP);}
">="                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","GE_OP"); return(GE_OP);}
"=="                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","EQ_OP"); return(EQ_OP);}
"!="                  { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","NE_OP"); return(NE_OP);}
";"                           { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","SEMI_COLON"); return(SEMI_COLON);}
("{"|"<%")            { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","LEFT_CURLY"); return(LEFT_CURLY);}
("}"|"%>")            { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","RIGHT_CURLY"); return(RIGHT_CURLY);}
","                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","COMMA"); return(COMMA);}
"="                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","ASGN_OP"); return(ASGN_OP);}
":"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","COLON"); return(COLON);}
"("                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","LEFT_PARENTEHSIS"); return(LEFT_PARENTHESIS);}
")"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","RIGHT_PARENTHESIS"); return(RIGHT_PARENTHESIS);}
("["|"<:")            { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","LEFT_BRACKET"); return(LEFT_BRACKET);}
("]"|":>")            { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","RIGHT_BRACKET"); return(RIGHT_BRACKET);}
"."                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","DOT"); return(DOT);}
"&"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","AMPERSAND"); return(AMPERSAND);}
"!"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","NOT_OP"); return(NOT_OP);}
"~"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","BTW_NOT"); return(BTW_NOT);}
"-"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","MINUS"); return(MINUS);}
"+"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","PLUS"); return(PLUS);}
"*"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","MUL_OP"); return(MUL_OP);}
"/"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","DIV_OP"); return(DIV_OP);}
"%"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","MOD_OP"); return(MOD_OP);}
"<"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","LESS_THAN"); return(LESS_THAN);}
">"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","GREATER_THAN"); return(GREATER_THAN);}
"^"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","XOR_OP"); return(XOR_OP);}
"|"                   { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","BTW_OR"); return(BTW_OR);}

[ \t\v\n\f]           { count();}
.                     { fprintf(fp,"<"); count(); fprintf(fp,",%s> ","INVALID"); return(INVALID);}

%%
```

```
int yywrap()
{
	return(1);
}


int column = 0;

void count()
{
	int i;

	for (i = 0; yytext[i] != '\0'; i++)
		if (yytext[i] == '\n')
			column = 0;
		else if (yytext[i] == '\t')
			column += 8 - (column % 8);
		else
			column++;
	fprintf(fp,"%s",yytext);
}
```