Exercise 9 - Implementation of code generation

Harshini S
185001058

Program Code:

gen.l

```
%{
	#include <stdlib.h>
	#include <stdio.h>
	#include "y.tab.h"
	void yyerror(char*);
%}

%%
[ \t]+
[0-9]+	{yylval=atoi(yytext);return NUMBER;}
[a-z]+	{yylval=*yytext-'a'; return ID;}
":="	{return ASSIGN;}
[-+*/]	{return *yytext;}
\n	{return *yytext;}
.	{char message[30];
	sprintf(message,"%s <%s>","Invalid character",yytext);
	yyerror(message);}
%%

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

gen.y

```
%{
	#include<stdio.h>
	#include<stdlib.h>
	#include<string.h>
	#include "y.tab.h"
	int yylex(void);
	void yyerror(char *);
	char *newTemp(char temp[]);
	char str1[7];
	extern FILE *yyin;
	struct attr	{
		char var[3];
		int num;
```

```
        };
%}
%token ID NUMBER ASSIGN
%left '+' '-'
%left '*' '/'

%%
program: program line'\n'|line'\n'
line:ID ASSIGN E'+'E {struct attr *n1,*n2,*n;
n = malloc(sizeof(struct attr));
n1 = malloc(sizeof(struct attr));
n2 = malloc(sizeof(struct attr));
n->var[0] = $1+97;
n->var[1] = '\0';
n1 = $3;
n2 = $5;
char temp[7];
newTemp(temp);
strcpy(str1, temp);
if(strcmp(n1->var,"")==0 && (strcmp(n2->var,"")!=0)){
printf("MOV %s, #%d\n",temp,n1->num);
printf("ADD %s, %s\n",temp,n2->var);
        }
        else if(strcmp(n1->var,"")!=0 && strcmp(n2->var,"")==0)      {
                printf("MOV %s, %s\n",temp,n1->var);
                printf("ADD %s, #%d\n",temp,n2->num);
                }
                else if(strcmp(n1->var,"")==0 && strcmp(n2->var,"")==0)     {
                        printf("MOV %s, #%d\n",temp,n1->num);
                        printf("ADD %s, #%d\n",temp,n2->num);
                }
                else    {
                        printf("MOV %s, %s\n",temp,n1->var);
                        printf("ADD %s, %s\n",temp,n2->var);
                        }
                printf("MOV %s, %s\n",n->var,temp);
                struct attr *result = malloc(sizeof(struct attr));
                strcpy(result->var, temp);
                $$ = result;
        }

|ID ASSIGN E'-'E{struct attr *n1,*n2,*n;
n = malloc(sizeof(struct attr));
n1 = malloc(sizeof(struct attr));
n2 = malloc(sizeof(struct attr));
n->var[0] = $1+97;
n->var[1] = '\0';
```

```
n1 = $3;
n2 = $5;
char temp[7];
newTemp(temp);
strcpy(str1, temp);
if(strcmp(n1->var,"")==0 && (strcmp(n2->var,"")!=0))
{
        printf("MOV %s, #%d\n",temp,n1->num);
        printf("SUB %s, %s\n",temp,n2->var);
}
else if(strcmp(n1->var,"")!=0 && strcmp(n2->var,"")==0)
{
printf("MOV %s, %s\n",temp,n1->var);
printf("SUB %s, #%d\n",temp,n2->num);
}
else if(strcmp(n1->var,"")==0 && strcmp(n2->var,"")==0)
{
printf("MOV %s, #%d\n",temp,n1->num);
printf("SUB %s, #%d\n",temp,n2->num);
}
else    {
printf("MOV %s, %s\n",temp,n1->var);
printf("SUB %s, %s\n",temp,n2->var);
}
printf("MOV %s, %s\n",n->var,temp);
struct attr *result = malloc(sizeof(struct attr));
strcpy(result->var, temp);
$$ = result;
}
|ID ASSIGN E'*'E
{struct attr *n1,*n2,*n;
n = malloc(sizeof(struct attr));
n1 = malloc(sizeof(struct attr));
n2 = malloc(sizeof(struct attr));
n->var[0] = $1+97;
n->var[1] = '\0';
n1 = $3;
n2 = $5;
char temp[7];
newTemp(temp);
strcpy(str1, temp);
if(strcmp(n1->var,"")==0 && (strcmp(n2->var,"")!=0))
{
printf("MOV %s, #%d\n",temp,n1->num);
printf("MUL %s, %s\n",temp,n2->var);
}
else if(strcmp(n1->var,"")!=0 && strcmp(n2->var,"")==0)
```

```
{
printf("MOV %s, %s\n",temp,n1->var);
printf("MUL %s, #%d\n",temp,n2->num);
}
else if(strcmp(n1->var,"")==0 && strcmp(n2->var,"")==0)
{
printf("MOV %s, #%d\n",temp,n1->num);
printf("MUL %s, #%d\n",temp,n2->num);
}
else    {
printf("MOV %s, %s\n",temp,n1->var);
printf("MUL %s, %s\n",temp,n2->var);
}
printf("MOV %s, %s\n",n->var,temp);
struct attr *result = malloc(sizeof(struct attr));
strcpy(result->var, temp);
$$ = result;
        }
|ID ASSIGN E'/'E{struct attr *n1,*n2,*n;
                n = malloc(sizeof(struct attr));
                n1 = malloc(sizeof(struct attr));
                n2 = malloc(sizeof(struct attr));
                n->var[0] = $1+97;
                n->var[1] = '\0';
                n1 = $3;
                n2 = $5;
                char temp[7];
                newTemp(temp);
                strcpy(str1, temp);
                if(strcmp(n1->var,"")==0 && (strcmp(n2->var,"")!=0))
                {
                printf("MOV %s, #%d\n",temp,n1->num);
                printf("DIV %s, %s\n",temp,n2->var);
                }
                else if(strcmp(n1->var,"")!=0 && strcmp(n2->var,"")==0)
                {
                printf("MOV %s, %s\n",temp,n1->var);
                printf("DIV %s, #%d\n",temp,n2->num);
                }
                else if(strcmp(n1->var,"")==0 && strcmp(n2->var,"")==0)    {
                printf("MOV %s, #%d\n",temp,n1->num);
                printf("DIV %s, #%d\n",temp,n2->num);
                }
                else    {
                        printf("MOV %s, %s\n",temp,n1->var);
                        printf("DIV %s, %s\n",temp,n2->var);
                        }
```

```
                        printf("MOV %s, %s\n",n->var,temp);
                        struct attr *result = malloc(sizeof(struct attr));
                        strcpy(result->var, temp);
                        $$ = result;
                        }

;
E:ID            {struct attr *n;
                        n = malloc(sizeof(struct attr));
                        n->var[0] = $1+97;
                        n->var[1] = '\0';
                        $$ = n;
}
|NUMBER               {struct attr *n;
                        n = malloc(sizeof(struct attr));
                        n->num = $1;
                        strcpy(n->var,"");
                        $$=n;
}
;
%%

char *newTemp(char temp[])
{
        static int count = 0;
        sprintf(temp,"R%d",count);
        count++;
        return temp;
}
void yyerror(char *s)   {
        fprintf(stderr,"%s",s);
        return;
}
int main()
{
        FILE *fp;
        printf("Input code : \n");
        fp = fopen("input.txt","r");
        char c = fgetc(fp);
        while(c!=EOF){
                printf("%c",c);
                c = fgetc(fp);
        }
        printf("\nAssembly code generated: \n");
        fclose(fp);
        fp = fopen("input.txt","r");
        yyin = fp;
```

```
        yyparse();
        fclose(fp);
        return 0;
}
```