Practical - 9

2CS701 – Compiler Construction

Harshit Gajipara
19BCE059

Aim:

To implement Assembly code generator; extend practical 6 to generate an assembly code. (use getReg() algorithm)

Code:

practical9.c

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>

typedef struct
{
    char var[10];
    int alive;
} regist;

regist preg[10];

void substring(char exp[], int st, int end)
{
    int i, j = 0;
    char dup[10] = "";
    for (i = st; i < end; i++)
        dup[j++] = exp[i];
    dup[j] = '0';
    strcpy(exp, dup);
}

int getregister(char var[])
{
    int i;
    for (i = 0; i < 10; i++)
    {
        if (preg[i].alive == 0)
```

```
        {
            strcpy(preg[i].var, var);
            break;
        }
    }
    return (i);
}

void getvar(char exp[], char v[])
{
    int i, j = 0;
    char var[10] = "";
    for (i = 0; exp[i] != '\0'; i++)
        if (isalpha(exp[i]))
            var[j++] = exp[i];
        else
            break;
    strcpy(v, var);
}

void main()
{
    char basic[10][10], var[10][10], fstr[10], op;
    int i, j, k, reg, vc, flag = 0;

    printf("\nEnter the Three Address Code:\n");
    for (i = 0;; i++)
    {
        gets(basic[i]);
        if (strcmp(basic[i], "exit") == 0)
            break;
    }

    printf("\nThe Equivalent Assembly Code is:\n");
    for (j = 0; j < i; j++)
    {
        getvar(basic[j], var[vc++]);
        strcpy(fstr, var[vc - 1]);
```

```
        substring(basic[j], strlen(var[vc - 1]) + 1,
strlen(basic[j]));
        getvar(basic[j], var[vc++]);

        reg = getregister(var[vc - 1]);

        if (preg[reg].alive == 0)
        {
            printf("\nMOV R%d,%s", reg, var[vc - 1]);
            preg[reg].alive = 1;
        }

        op = basic[j][strlen(var[vc - 1])];
        substring(basic[j], strlen(var[vc - 1]) + 1,
strlen(basic[j]));
        getvar(basic[j], var[vc++]);

        switch (op)
        {
            case '+': printf("\nADD"); break;
            case '-': printf("\nSUB"); break;
            case '*': printf("\nMUL"); break;
            case '/': printf("\nDIV"); break;
        }

        flag = 1;
        for (k = 0; k <= reg; k++)
        {   if (strcmp(preg[k].var, var[vc - 1]) == 0)
            {
                printf("R%d, R%d", k, reg);
                preg[k].alive = 0;
                flag = 0;
                break;
            }
        }

        if (flag)
        {
```

```
            printf(" %s,R%d", var[vc - 1], reg);
            printf("\nMOV %s,R%d", fstr, reg);
        }

        strcpy(preg[reg].var, var[vc - 3]);
        getch();
    }
    printf("\n\n");
}
```

Output:

```
Enter the Three Address Code:
p=q+r
exit

The Equivalent Assembly Code is:

MOV R0,q
ADD r,R0
MOV p,R0
```

```
Enter the Three Address Code:
a=b/c
exit

The Equivalent Assembly Code is:

MOV R0,b
DIV c,R0
MOV a,R0
```

```
Enter the Three Address Code:
s=t*p
exit

The Equivalent Assembly Code is:

MOV R0,t
MUL p,R0
MOV s,R0
```

```
Enter the Three Address Code:
a=b*c
b=d/e
c=d-e
d=e*f
exit

The Equivalent Assembly Code is:

MOV R0,b
MUL c,R0
MOV a,R0
MOV R1,d
DIV e,R1
MOV b,R1
MOV R2,d
SUB e,R2
MOV c,R2
MOV R3,e
MUL f,R3
MOV d,R3
```