代码生成作业

丁元杰 17231164

2019 年 11 月 12 日

15.1

(1)

栈式：

```
PUSH B
PUSH C
ADD
PUSH A
MUL
POP T1
PUSH B
PUSH C
ADD
POP T2
PUSH D
PUSH T2
DIV
POP T2
PUSH T1
PUSH T2
SUB
POP X
```

累加式：

```
LOAD B
ADD C
STORE T1
LOAD A
MUL T1
STORE T1
LOAD B
```

```
ADD C
STORE T2
LOAD D
DIV T2
STORE T2
LOAD T1
SUB T2
LOAD X
```

寄存器-内存式：

```
LOAD R0, B
ADD R1, R0, C
MUL R1, A, R1
LOAD R2, B
ADD R2, R2, C
DIV R3, D, R2
SUB R0, R1, R3
STORE R0, X
```

寄存器-寄存器式：

```
LOAD R0, B
LOAD R1, C
ADD R0, R0, R1
LOAD R1, A
MULT R0, R0, R1
LOAD R1, B
LOAD R2, C
ADD R1, R1, R2
LOAD R2, D
DIV R2, R2, R1
SUB R0, R0, R2
STORE R0, X
```

(2)

栈式：$3 \times 18 = 54$周期
累加式：$3 \times 15 = 45$周期
寄存器-内存式：$3 \times 7 + 1 \times 1 = 22$周期
寄存器-寄存器式：$3 \times 7 + 1 \times 5 = 26$周期

15.4

如果分配在静态区，那么执行结果将会是：

```
0
1
```

而如果分配在栈上，执行结果变成：

```
0
0
```

原因是C语言中，栈上分配的局部变量会在每次调用时分配内存，同时初始化，因此每次都会被清零；而分配在静态区的变量，其生命周期与程序同长，因而只有一次初始化，第二次使用时则直接引用静态区存储的值。

15.5

分配过程：

1. 选择b，因为它的度数小于2
2. 选择x，指定不为其分配寄存器
3. 选择a，指定不为其分配寄存器
4. 选择y，将其移走，因为度数小于2。现在只剩一个结点i
5. 将i分配在寄存器$R0$上
6. 将y加入图中，分配在寄存器$R1$上
7. 将b加入图中，分配在寄存器$R0$上

最后的分配结果：

- i, b，使用$R0$寄存器
- y，使用$R1$寄存器
- a, x，不使用全局寄存器

15.6

四元式

```
T1 = B + C
T2 = A * T1
T3 = B + C
```

```
T4 = D * T3
T5 = T2 - T4
```

汇编代码：

```
mov     edx, DWORD PTR [esp + 10]
mov     eax, DWORD PTR [esp + 14]
add     eax, edx
imul    eax, DWORD PTR [esp + 18]
mov     edx, eax
mov     ecx, DWORD PTR [esp + 10]
mov     eax, DWORD PTR [esp + 14]
add     eax, ecx
imul    eax, DWORD PTR [esp + 1C]
sub     edx, eax
mov     eax, edx
```