HW4

4.13

4.13 语句的文法如下：

$S \rightarrow$ **id** $:=E$ | **if** E **then** S | **while** E **do** S | **begin** $S;S$ **end** | **break**

写一个翻译方案，其语义动作的作用是：若发现 **break** 不是出现在循环语句中，及时报告错误。

loop是S的一个继承属性

$S' \rightarrow \{S.loop = false;\}S$

$S->id := E$

$S->if \quad E \quad then\{S_1.loop = S.loop;\}S_1$

$S->while \quad E \quad do\{S_1.loop = true;\}S_1$

$S->begin \quad \{S_1.loop = S.loop;\}S_1;\{S_2.loop = S.loop;\}S_2 \quad end$

$S->break \quad \{if(!S.loop) \quad print(\text{"error"});\}$

4.15

4.15 下面是构造语法树的一个 S 属性定义。将这里的语义规则翻译成 LR 翻译器的栈操作代码段。

$E \rightarrow E_1 + T$	$E.nptr = mkNode('+', E_1.nptr, T.nptr)$
$E \rightarrow E_1 - T$	$E.nptr = mkNode('-', E_1.nptr, T.nptr)$
$E \rightarrow T$	$E.nptr = T.\ nptr$
$T \rightarrow (E)$	$T.nptr = E.\ nptr$
$T \rightarrow$ **id**	$T.nptr = mkLeaf($ **id**, **id**$.entry)$
$T \rightarrow$ **num**	$T.nptr = mkLeaf($ **num**, **num**$.val)$

代码段等号左部为val[top-r+1],r为待归约的产生式右部符号数，分别为3、3、1、3、1、1

产生式	代码段
$E \rightarrow E_1 + T$	$val[top-2] = mknode('+', val[top-2], val[top])$
$E \rightarrow E_1 - T$	$val[top-2] = mknode('-', val[top-2], val[top])$
$E \rightarrow T$	
$T \rightarrow (E)$	$val[top-2] = val[top-1]$
$T \rightarrow id$	$val[top] = mkleaf(id, val[top])$
$T \rightarrow num$	$val[top] = mkleaf(num, val[top])$

5.5

5.5 假如有下列 C 的声明：

```
typedef struct {
        int a,b;
} CELL, * PCELL;
CELL foo[100];
PCELL bar(x,y) int x;CELL y;{…}
```

为变量 foo 和函数 bar 的类型写出类型表达式。

foo的类型表达式：$array(0..99, record((a \times integer) \times (b \times integer)))$

bar的类型表达式：

$(integer \times record((a \times integer) \times (b \times integer))) \rightarrow pointer(record((a \times integer) \times (b \times integer)))$

6.5

*6.5 一个C语言程序如下：

```
typedef struct_a{
    short  i;
    short  j;
    short  k;
}a;
typedef struct_b{
    long   i;
    short  k;
}b;
main(){
    printf("Size of short, long,a and b=%d,%d,%d,%d\n",
        sizeof(short),sizeof(long),sizeof(a),sizeof(b));
}
```

该程序在Ubuntu 12.04.2 LTS(GNU/Linux 3.2.0-42-generic x86_64)系统上，经过编译器GCC:(Ubuntu/Linaro 4.6.3-1ubuntu5)4.6.3编译后，运行结果如下：

```
Size of short,long,a and b=2,8,6,16
```

已知short类型和long类型分别对齐到2的倍数和8的倍数。试问，为什么类型b的size会等于16?

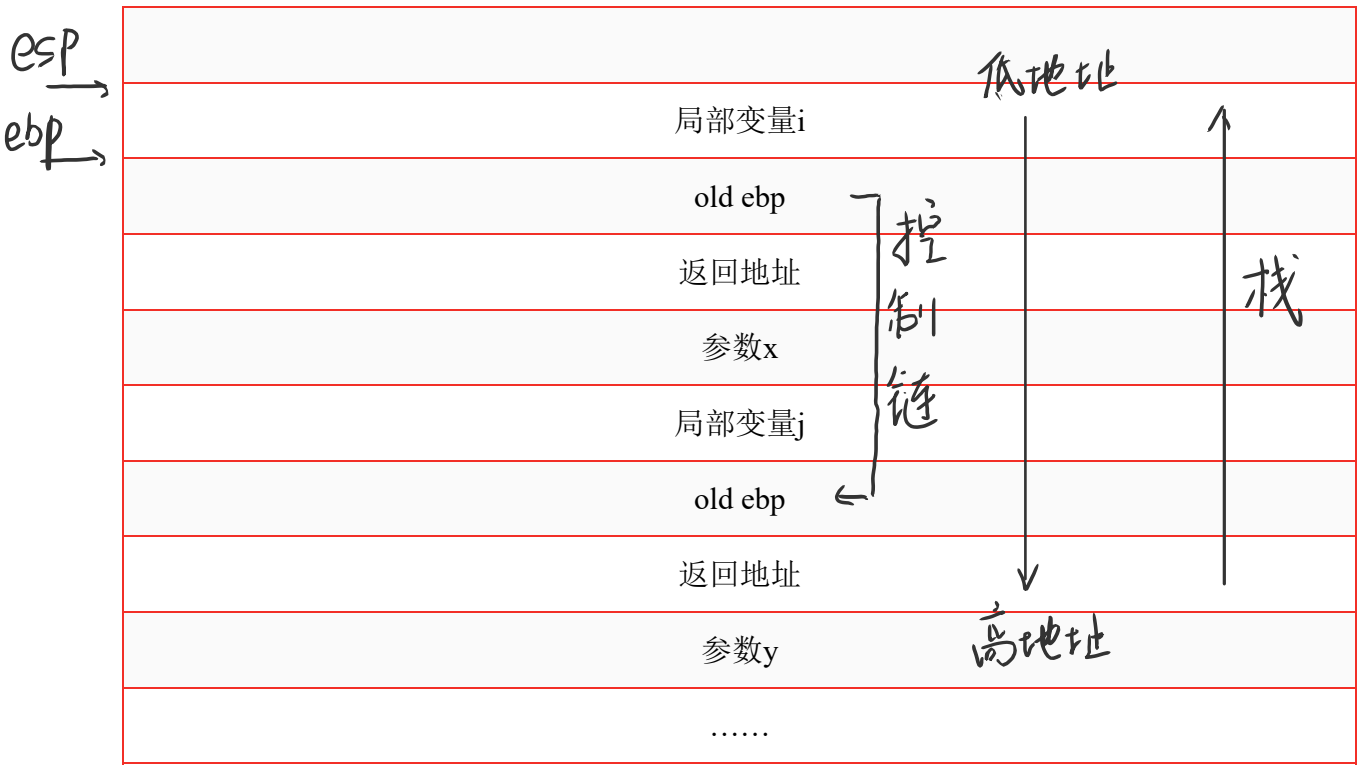
结构体中可以存放不同类型的数据，但是大小并不是简单的各个类型之和，由于读取内存的要求，例如在本题中，long是对齐8的倍数，当结构体b以数组形式出现时，若只是按照简单相加求和为10，为了保证long类型元素的对齐，只能数组元素之间空6个字节，但是这样不满足数组size的计算原则（元素个数×元素size），因此整个结构体的大小应该是最大对齐数的整数倍，本题中有两个元素，则总大小为8*2=16

6.6

6.6 下面是C语言两个函数f和g的概略(它们不再有其他的局部变量)：

```
int f(int x){int i;…return i +1;…}
int g(int y){int j;…f(j+1);…}
```

请按照图6.11的形式，画出函数g调用f,f的函数体正在执行时，活动记录栈的内容及相关信息，并按图6.10左侧箭头方式画出控制链。假定函数返回值是通过寄存器传递的。


esp
ebp
局部变量i
old ebp
返回地址
参数x
局部变量j
old ebp
返回地址
参数y
……
低地址
高地址
控制链
栈