Compiler Code Generations
Appendix for Andes

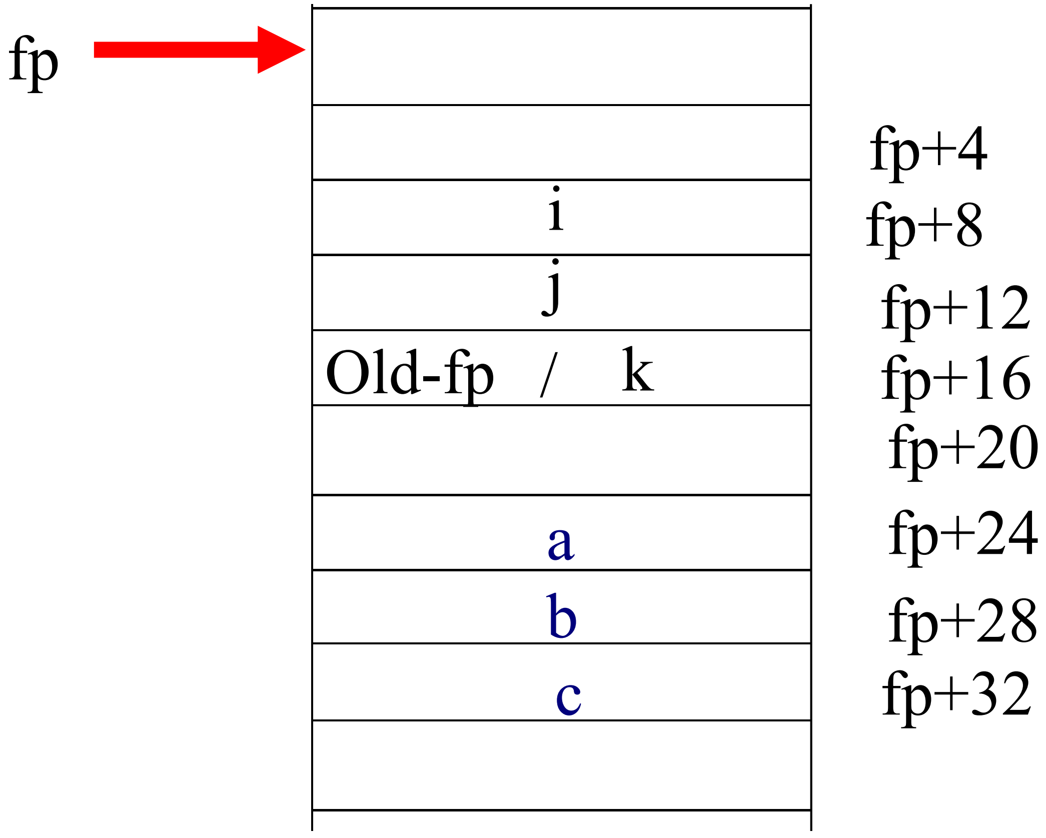
Jenq Kuen Lee
Department of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan

Variable Naming

- Argument list.
- Local variables.

```
foo (int a, int b, int c){
int i, j, k;

  i = a + b + 3 +6;
  a= i+ j+k+3;
}
```



Four items in function call handlings

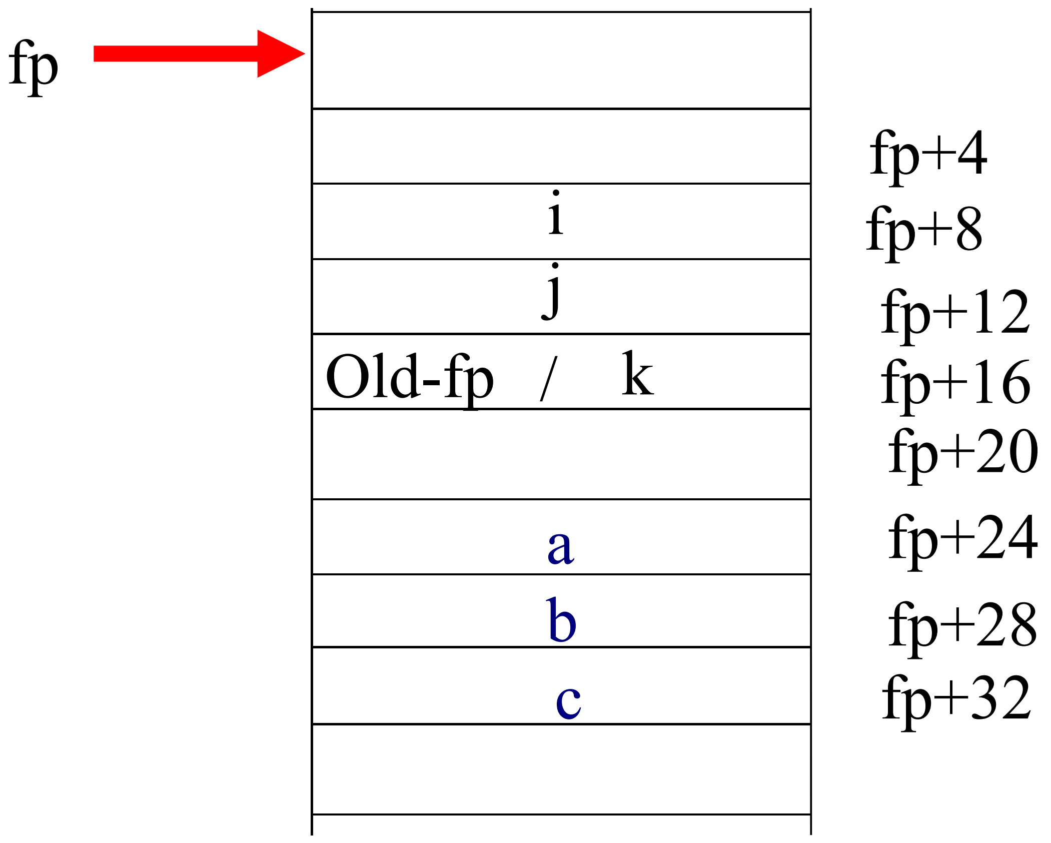
- A. Enter function body (callee).
- B. Exit function body (callee).
- C. Function invocation.
- D. Return from function invocation to caller.

```
*/Enter and Exit function
   Body*/
 int foo (int i, j,k){
   /* Section A */
    int a, b, c;

   /* Section B */
    return (a+3);
}
```

```
/* Function Call */
  /* Section  C */
       = Foo (a1,a2,a3);
  /* Section  D */
  X
```

A. Entering function body

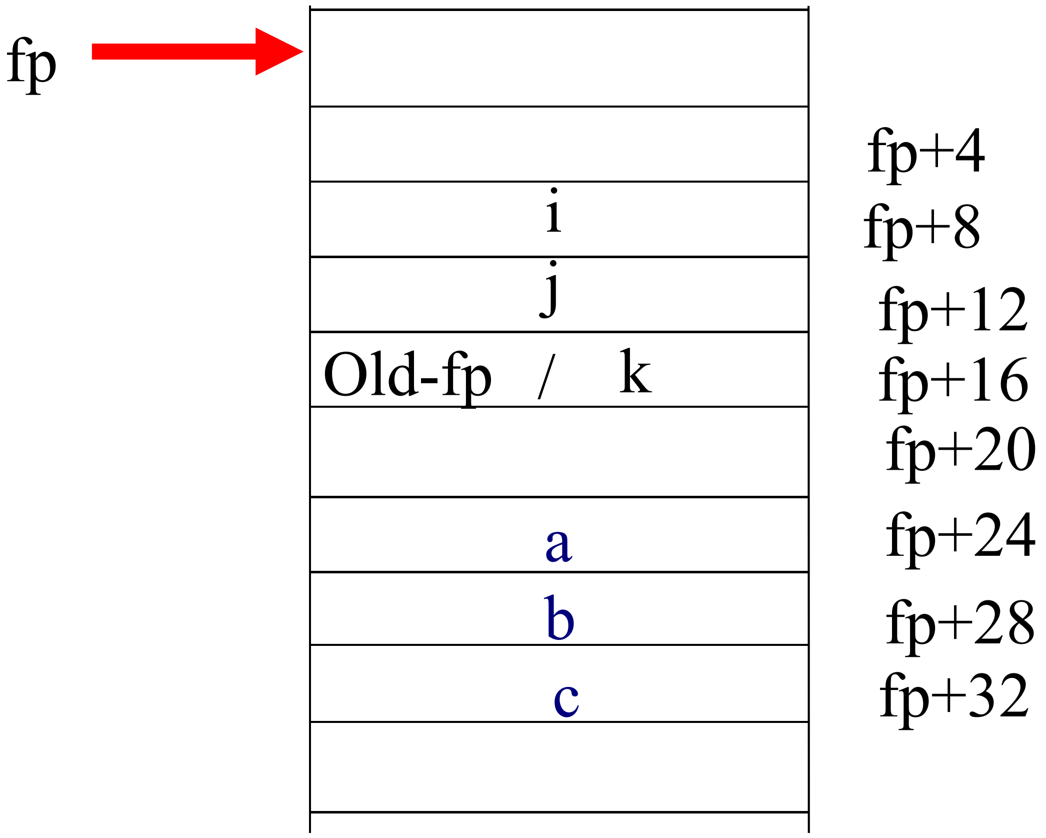


```
foo (int a, int b, int c){
int i, j, k;
  /* Section A */
   i = a + b + 3 +6;
   a= i+ j+k+3;
}
```

```
/* Section A */

addi    $sp, $sp, -20
swi     $fp, [$sp], -4
addi    $fp, $sp, 0
sp=sp+SizeOf (LocalVars)
```

B. Exit function body

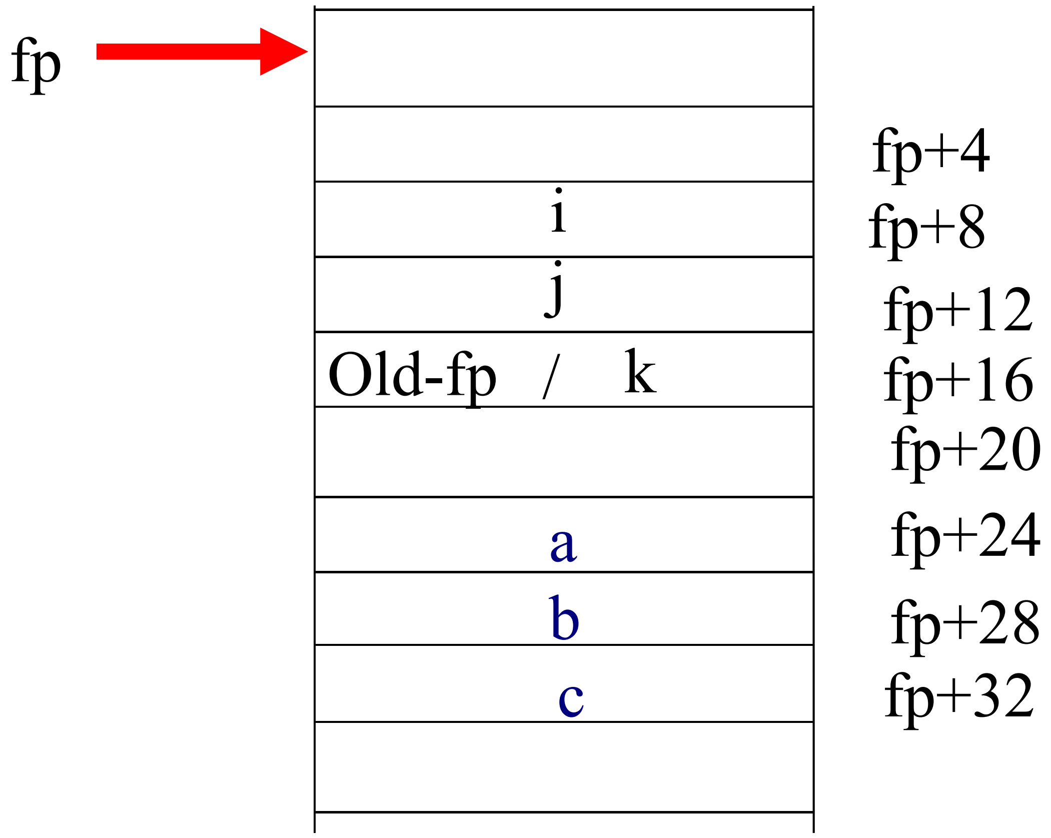


```
foo (int a, int b, int c){
int i, j, k;
  i = a + b + 3 +6;
   a= i+ j+k+3;
   /* Section B */
   return (a+2);
}
```

```
/* Section B */

addi    $sp, $fp, 0
lmw.aim $sp, [$sp], $sp, 8
addi    $sp, $sp, 20
ret
```

C. Function invocation



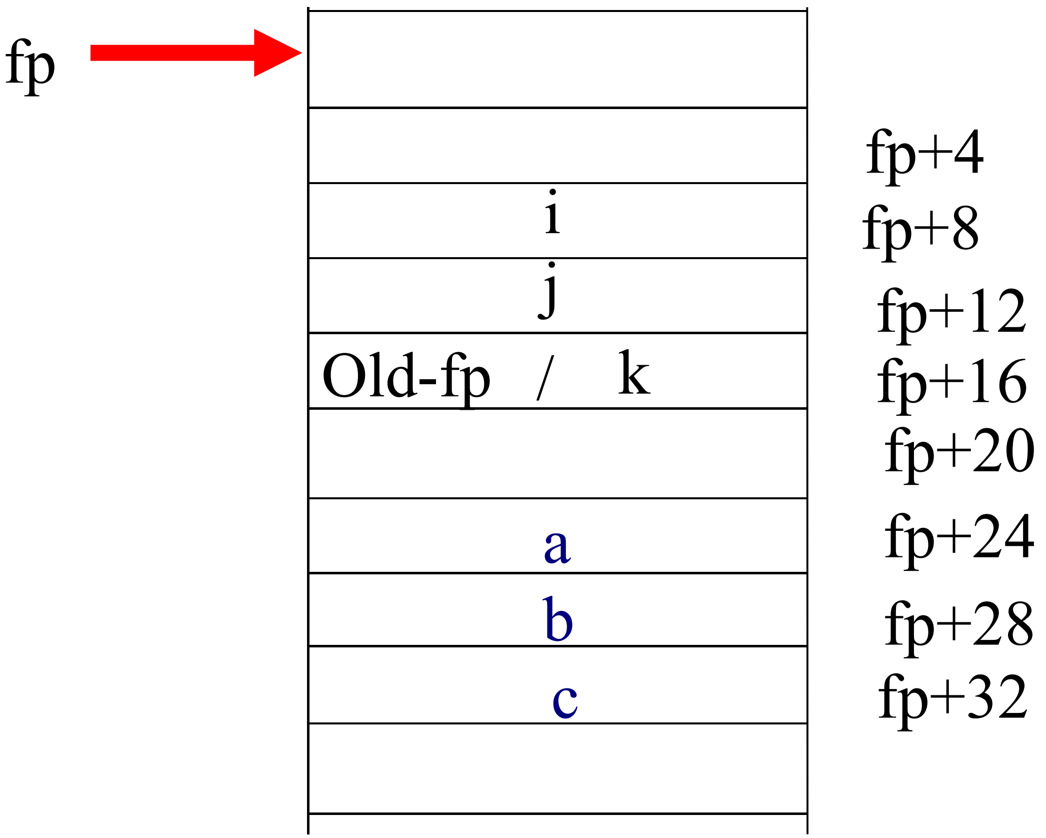
```
/* Function Call */
   /* Section C */
        = Foo (a1,a2,a3);
   /* Section D */
   X
```

```
/* Section C */
 movi   $r0, 1
 movi   $r1, 2
 movi   $r2, 3



 bal    foo
```

D. Return from function invocation to caller



```
/* Function Call */

= Foo (a1,a2,a3);
   /* Section  D */
   X
```

```
/* Section D */

swi    $r0, [$fp+(8)];
/*return value*/
```

Overview of Routines

```
extdef:
      TYPESPEC notype_declarator ';'
       { if (TRACEON) printf("7 ");
           set_global_vars($2);
         }
       | notype_declarator
               { if (TRACEON) printf("10 ");
                 cur_scope++;
                 set_scope_and_offset_of_param($1);
                code_gen_func_header($1);
                }
         '{'  xdecls
               { if (TRACEON) printf("10.5 ");
                 set_local_vars($1);
                }
              stmts
              {
                if (TRACEON) printf("11 ");
                pop_up_symbol(cur_scope);
                cur_scope--;
     code_gen_at_end_of_function_body($1);
              }
```

Overview of Routines

- Adjust parameter scopes.

```
/* Set up parameter scope and offset */
set_scope_and_offset_of_param(char *s) {
  int i,j,index;
  int total_args;
   index = look_up_symbol(s);
   if (index<0) err("Error in function header");
   else {
      table[index].type = T_FUNCTION;
      total_args = cur_counter -index -1;
      table[index].total_args=total_args;
      for (j=total_args, i=cur_counter-1;i>index; i--,j--)
        {
          table[i].scope= cur_scope;
          table[i].offset= j;
          table[i].mode  = ARGUMENT_MODE;
        }
   }
}
```

Overview of Routines

- Entering function body.

```
/* To generate house-keeping work at the
    beginning of the function */
code_gen_func_header(functor)
char *functor;
{

fprintf(f_asm,"        .text\n");
fprintf(f_asm,"        .globl   %s\n",functor);
fprintf(f_asm,"%s:\n",functor);
fprintf(f_asm,"        addi    $sp, $sp, -20\n");
fprintf(f_asm,"        swi     $fp, [$sp], -4\n");
fprintf(f_asm,"        addi    $fp, $sp, 0\n");
}
```

Overview of Routines

- Exit function body.

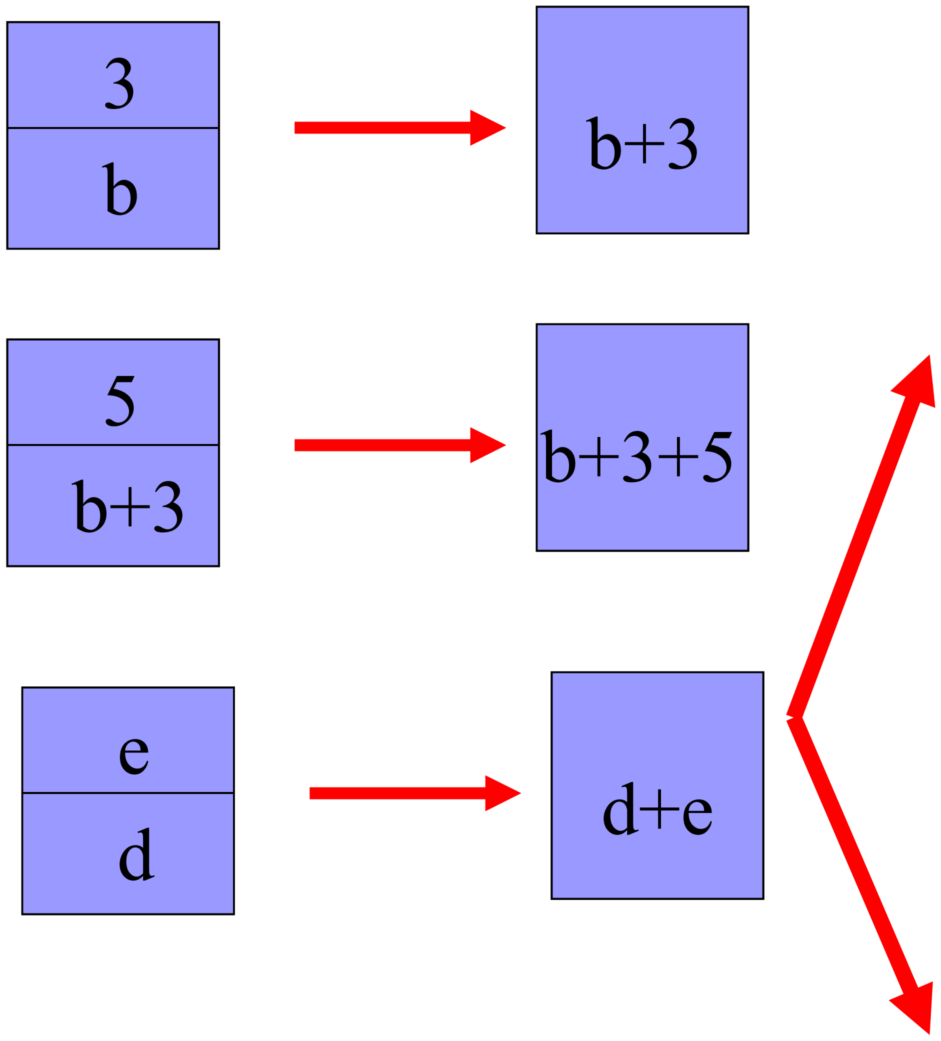
```
/* To generate house-keeping work at the
    end of a function */

code_gen_at_end_of_function_body(func
    tor)
char *functor;
{
  int i;

  fprintf(f_asm,"       addi    $sp, $fp, 0\n");
  fprintf(f_asm,"       lmw.aim $sp, [$sp],
    $sp, 8\n");
  fprintf(f_asm,"       addi    $sp, $sp,20\n");
  fprintf(f_asm,"       addiu  $sp, 8\n");
  fprintf(f_asm,"       ret\n");
}
```

Expression Code Gen:

- Stack Operation: b+3+5



X86 Assembly

```
push d
push e
pop bx
pop cx
add bx,cx
push bx
```

Expression Grammars

```
expr_no_commas:  primary
          { if (TRACEON) printf("15 ") ;
            $$= $1;
          }
     | expr_no_commas '+' expr_no_commas
             {
                if (TRACEON) printf("16 ") ;
                fprintf(f_asm,"       lwi    $r1, [$fp+(24)])\n");
                fprintf(f_asm,"       lwi    $r1, [$fp+(28)])\n");
                fprintf(f_asm,"       add   $r0, $r1, $r0\n");
                fprintf(f_asm,"       swi     $r0, [$fp+(8)]\n");
                fprintf(f_asm,"       addiu $sp, 4\n");
              $$= NULL;
              }
     | expr_no_commas '=' expr_no_commas
             { ...}
      | expr_no_commas '*' expr_no_commas
             {...}
             ;
```

Expression Grammars(2

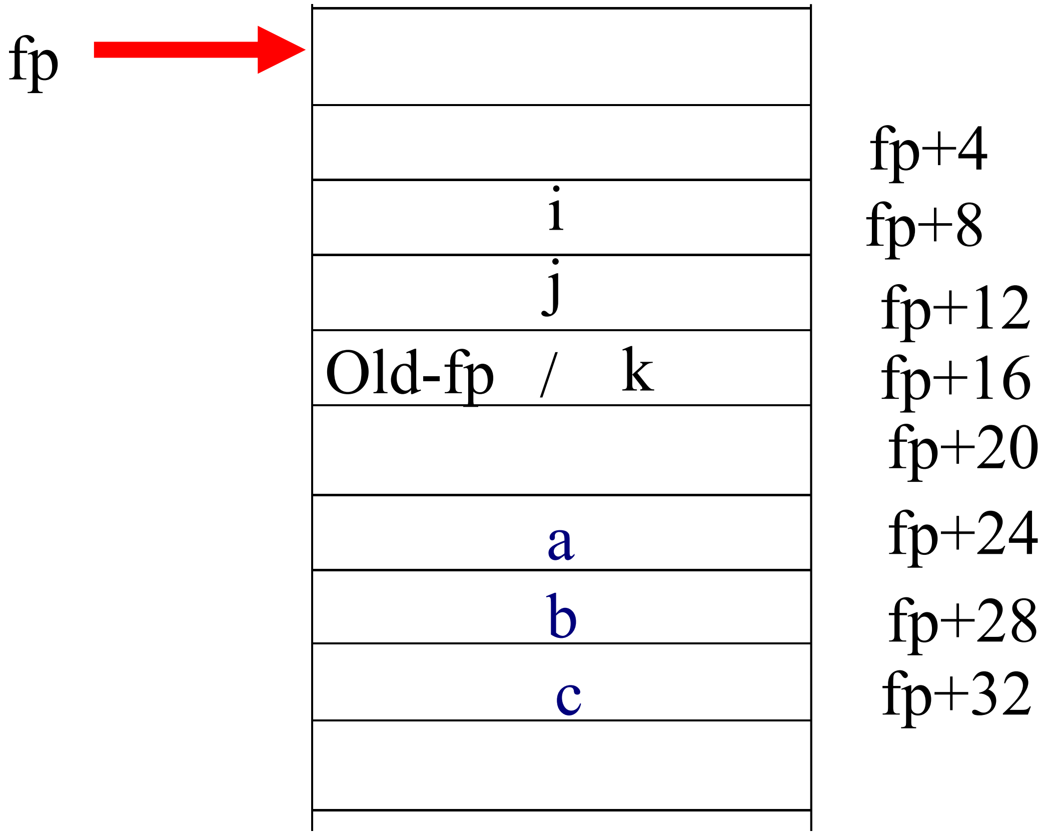
```
expr_no_commas:  primary
          { if (TRACEON) printf("15 ") ;
             $$= $1;
          }
      | expr_no_commas '+' expr_no_commas
                { . . .}
      | expr_no_commas '=' expr_no_commas
                {  . . .}
      | expr_no_commas '*' expr_no_commas
      { if (TRACEON) printf("18 ") ;
        fprintf(f_asm,"        lwi   $r1,[$fp+(24)]\n");
        fprintf(f_asm,"        lwi    $r1,[$fp+(24)] \n");
        fprintf(f_asm,"        mul  $r0, $r1, $r0\n");
        fprintf(f_asm,"        swi   $r0, [$fp+(8)]\n");
         $$= NULL;
       }                  ;
```

Variable Naming

- Argument list.
- Local variables.

```
foo (int a, int b, int c){
int i, j, k;

   i = a + b + 3 +6;
   a= i+ j+k+3;
}
```



Expression Assign:

X = 3;

```
expr_no_commas:  . . . { . . .}
      | expr_no_commas '=' expr_no_commas
    { char *s;    int index;
              s= $1;  if (!s) err("improper expression at LHS");
              index = look_up_symbol(s);
      switch(table[index].mode) {
      case ARGUMENT_MODE:
  fprintf(f_asm,"       swi   $r0,
      [$fp+(%d)]\n",table[index].offset*4+4);
      case  LOCAL_MODE:
               fprintf(f_asm,"       movi    $r0,3" );
               fprintf(f_asm,"       swi   $r0,
      [$fp+(%d)]\n",table[index].offset*4+4);
              default: /* Global Vars */
         fprintf(f_asm," sethi   $r0, hi20(%s)\n", table[index].name);
               fprintf(f_asm,"       lwi    $r0, [$r0+lo12(%s)]\n");
       }  } . . . ;
```

Primary Expr.

```
a + b+ 3;
```

```
primary: IDENTIFIER  {
            int index;
              index =look_up_symbol($1);
              switch(table[index].mode) {
    case ARGUMENT_MODE:
             fprintf(f_asm,"        lwi    $r1, [$fp+(%d)]\n"
             ,table[index].offset*4+4);
    break;
    case LOCAL_MODE:
              fprintf(f_asm,"        lwi    $r1, [$fp+(%d)]\n"
              ,table[index].offset*4+4);
     break;
    default: /* Global Vars */
    fprintf(f_asm," sethi   $r0, hi20(%s)\n", table[index].name);
    fprintf(f_asm,"        lwi    $r0, [$r0+lo12(%s)]\n");
     }
              $$=$1;  }
    ;
```

Constant Expr.

```
primary:   IDENTIFIER  {            . . . }
    | CONSTANT
              { if (TRACEON) printf("21 ") ;
        fprintf(f_asm,"        movi  $r0 ,4);
        fprintf(f_asm,"        swi    $r0, [$fp+(8)]\n");
          }
| STRING
            {
             if (TRACEON) printf("22 ") ;
              }
    | primary PLUSPLUS
            {
             if (TRACEON) printf("23 ") ;
              }
 ;
```

Assignment3/ test1.c

```c
int a;
int b;

main()
{

   a= a+ 3;

}
```

```
smw.adm $sp, [$sp], $sp, 8
addi    $sp, $sp, -4
addi    $fp, $sp, 0
! end of prologue
sethi   $r0, hi20(a)
lwi     $r0, [$r0+lo12(a)]
addi    $r1, $r0, 3
sethi   $r0, hi20(a)
swi     $r1, [$r0+lo12(a)]
! epilogue
addi    $sp, $fp, 4
lwi.bi  $fp, [$sp], 4
ret
```

Assignment3/ test2.c

```c
int a;
int b;

main()
{
   int i;
   int j;
   int k;

   a = a + b + 3 + 6;
   a = i + j + k + 3;
}
```

```
# identifier
#For variable a
    sethi   $r0, hi20(a)
    lwi     $r1,[$r0+lo12(a)]

# identifier
#For variable b
    sethi   $r0, hi20(b)
    lwi     $r1,[$r0+lo12(b)]

# '+' operation
#For a=a+b+3+6
        add     $r0, $r1, $r0
        addi    $r1, $r0, 9
        sethi   $r0, hi20(a)
   swi     $r1, [$r0+lo12(a)]
```

```
# '+' operation
#For  a=i+j+k+3
   lwi     $r1, [$fp+(8)]
   lwi     $r0, [$fp+(12)]
   add     $r1, $r1, $r0
   lwi     $r0, [$fp+(16)]
   add     $r0, $r1, $r0
   addi    $r1, $r0, 3
   sethi   $r0, hi20(a)
   swi     $r1, [$r0+lo12(a)]

…
```