Code Generation Example

Tsan-sheng Hsu

tshsu@iis.sinica.edu.tw

`http://www.iis.sinica.edu.tw/~tshsu`

Warning

- This set of slides contain a “pseudo code” for a very simple compiler.
- This example does not pass LEX, YACC or GCC.
- Usage of this example is entirely to illustrate the high level ideas.
- Not responsibile for any syntax errors.
- Please read LEX, YACC and GCC manuals carefully to avoid programming mistakes.
    - http://dinosaur.compilertools.net/
    - school library

Error handling

```
char *error_message[MAXMSG]; /* a big error message array */
void error_msg(int line_no; char * file_name;
int index; char *msg1; char *msg2)
{
switch index {
           /* some messages needed to be specially treated */
           case 3: /* insert *msg1 into error message */
                 /* variable ‘‘name’’ undefined %/
                 printf("line %d in file %s, error %d, %s\n",
                 line_no,file_name,index,error_message[index]);
           ...
           default:
              if(index >= MAXMSG){
                 printf("internal error, wrong error index  %d\n",inde
              }else{
                 printf("line %d in file %s, error %d, %s\n",
                 line_no,file_name,index,error_message[index]);
      }       exit(1);}
```

Type Definitions

```
typedef struct place_typ {
         char tag;
         int offset;
} PLACE_TYPE;
typedef struct var_typ {
         PLACE_TYPE place;
         char *name;
         ...
} VAR_TYPE;
```

Emit

```
/* op: operator, opni: operand # i */
void emit(char operator; PLACE_TYPE opn1,opn2,opn3)
{
     switch operator {
       case ’+’:
            gen_r_address(opn2,3); /* load into R__3 */
            gen_r_address(opn3,4); /* load into R__4 */
            fprintf(code_file,"R__3 =  R__3+ R__4\n");
            gen_l_address(opn1,3);
            break;
       case ’-’: ...
       case ’*’: ...
       case ’/’: ...
       default:
             printf("internal error in code generation, operator
             %c is undefined\n",operator); exit(1);
      }
}
```

Generate R-address (1/2)

```
#define GLOBAL_VAR 1
#define LOCAL_VAR 2
#define TEMP_VAR 3
#define REG_VAR 4
#define PARA_VAR 5
#define NON_LOCAL_VAR 6
#define CONSTANT_VAR 7
...

int global_start; /* the place where you start to put global variables
int local_start; /* the place where you start to put local variables *
int temp_start; /* the place where you start to put temp variables */
```

Generate R-address (2/2)

```
/* load the value at ‘‘where’’ into register # result_r */
void gen_r_address(PLACE_TYPE where, int result_r)
{ switch where.tag {
case GLOBAL_VAR:
    fprint(code_file,"R__1 = TOP__S();\n");
    fprint(code_file,"R__1 = 0 - R__1;\n");
    fprint(code_file,"R__1 = R__1 + %d;\n",global_start+where.offset);
    fprint(code_file,"R__%1d = VAL__S(R__1);\n",result_r); break;
case LOCAL_VAR:
    fprint(code_file,"R__1 = TOP__S();\n");
    fprint(code_file,"R__1 = 0 - R__1;\n");
    fprint(code_file,"R__1 = R__1 + %d;\n",local_start+where.offset);
    fprint(code_file,"R__%1d = VAL__S(R__1);\n",result_r); break;
case CONSTANT_VAR:
    fprint(code_file,"R__%1d = %d;\n",result_r,where.offset); break;
case TEMP_VAR: ... } }
```

Generate L-address

```
/* store the value at register # result_r into the place ‘‘where’’ */
void gen_l_address(PLACE_TYPE where, int result_r)
{ switch where.tag {
        case GLOBAL_VAR:
          fprint(code_file,"R__1 = TOP__S();\n");
          fprint(code_file,"R__1 = 0 - R__1;\n");
          fprint(code_file,"R__1 = R__1 + %d;\n",global_start+where.of
          fprint(code_file,"SSET__S(R__1,R__%1d);\n",result_r);
          break;
        case LOCAL_VAR:
          fprint(code_file,"R__1 = TOP__S();\n");
          fprint(code_file,"R__1 = 0 - R__1;\n");
          fprint(code_file,"R__1 = R__1 + %d;\n",local_start+where.off
          fprint(code_file,"SSET__S(R__1,R__%1d);\n",result_r);
          break;
        case TEMP_VAR:
          ...
        default: /* internal error */ }  }
```

YACC service routines

```
char temp_map[MAXTEMP]; /* initialize to all zero */
int max_temp_used=0; /* max number of temps used */

void freetemp(PLACE_TYPE vplace)
{        if(vplace.tag == TEMP_VAR){
           temp_map[vplace.offset] = 0;
}

/* very simple allocation algorithm, first fit */
int newtemp()
{       int i=0;
        while(i < MAXTEMP && temp_map[i] > 0)  i++;
        if(i >= MAXTEMP){ /* internal error, allocate more temps */
        }else{
            if(i > max_temp_used) max_temp_used = i;
            return(i);
        }
}
```

YACC Rules

```
%union{
    int ival;
    double fval;
    VAR_TYPE vval;
}
%type <vval> expr /* define the return type of ‘‘expr’’ to be VAR_TYPE
...
expr : expr ’+’ expr
          {
             $$.place.offset = newtemp();  $$.place.tag = TEMP_VAR;
             emit(’+’,$$.place,$1.place,$2.place);
             freetemp($1.place);     freetemp($2.place);
          }
       | expr ’-’ expr   {...}
       ...
       | variable {...}
...
```