C−− Language V2.0

Tsan-sheng Hsu

tshsu@iis.sinica.edu.tw

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

Definition

- The C−− language is a subset of the standard C language.
- Its purpose is to act like a universal intermediate language.
- C−− is a STACK based language.
- A C−− program consists of the following parts.
  - #define MAX_S maximum_stack_size
    - *Allocate the size of the STACK.*
    - *Each STACk element can hold an integer or a float. That is, we assume the sizes of an integer and a float are the same.*
  - #include "cmm.c"
    - *this line is required and the file "cmm.c" contains system defined functions and variables.*
  - procedure_1()
  - procedure_2()
  - ...
  - procedure_n()

Procedure Definition

- **Each procedure_i is a standard C procedure without parameters.**
  - **procedure_i()**
  - **{**
  - **...**
  - **}**
- **Procedure_1 must be *main*.**
- **The first statement of *main* is INIT__S();**
- **Inside each procedure, the followings rules are enforced.**
  - **No variable declaration is allowed.**
  - **Constants are zero, positive or negative integers / floats.**
  - **Ten global $sizeof(int)$-byte integer registers.**
    - ***They are R__0, . . . , R__9.***
    - ***These variables are called* integer registers, *or I_register.***
  - **Ten global $sizeof(float)$-byte floating point registers.**
    - ***They are F__0, . . . , F__9.***
    - ***These variables are called* float registers, *or F_register.***

Statements

- Each line contains exactly one statement.
- Null statement — blank lines containing white spaces.
- Comments of the form
  - $/*\cdots*/$
- STACK oriented operations.
- Assignment statements.
- A C label of the form
  - *label:*
- Jump statements.
- I/O statements.
- Procedure call statements
  - *procedure_i();*

STACK operations

- **INIT__S();**
  - used only in the first statement of $main$.
  - Initialize the stack.
- **I_register = TOP__S();**
  - returns the current stack pointer.
  - Initial value is 0.
- **I_register = VAL__S(i); F_register = FVAL__S(i);**
  - returns the value at stack pointer $+i$.
- **SETSP__S(i);**
  - set new stack pointer to be current stack pointer $+i$.
- **SSET__S(i,k); FSSET__S(i,k);**
  - set the value at stack pointer $+i$ to k.
- **PUSH__S(k); FPUSH__S(k);**
- **I_register = POP__S(); F_register = FPOP__S();**
- **Note that i and k are registers or constants.**

Assignment statements

- **register = (register | constant) $(+|-|*|/|\%)$ (register | constant);**
  - ▷ ***No type conflict is allowed.***
- **left shift or right shift**
  - **Only for integers.**
  - **l_register <<= (l_register | constant);**
  - **l_register >>= (l_register | constant);**
- **l_register = (l_register | constant) $(\&, \hat{}, |)$ (l_register | constant);**
  - **Only for integers.**
  - **bit-wise AND, XOR and OR.**
- **register = (register | constant);**
  - ▷ ***No type conflict is allowed.***

Jump statements

- **Conditional jump**
  - **if '(' (I_register | I_constant)** $(> \mid < \mid == \mid >= \mid <=)$ **0 ')' goto label;**
  - **if '(' (F_register | F_constant)** $(> \mid < \mid == \mid >= \mid <=)$ **0.0 ')' goto label;**
- **Unconditional jump**
  - **goto label;**

I/O statements

- **Read an integer / a float into a register**
  - **scanf("%d",&I_register);**
  - **scanf("%f",&F_register);**
- **Print an integer / a float that is stored in a register**
  - **printf("%d",I_register);**
  - **printf("%f",F_register);**
- **Print a string**
  - **printf("string");**
- **Print a newline**
  - **printf("\n");**

A Sample C−− program

```
#define MAX__S 10000
#include "cmm.c"
main()
{
        INIT__S();
        R__0 = 1;
        scanf("%d",&R__1);
        if(R__1 <= 0) goto done;
        PUSH__S(R__1);
/* compute factorial */
        factorial();
compute:
        R__1 = POP__S();
        R__1 = R__1 - 2;
        if(R__1 <= 0) goto done;
        PUSH__S(R__1);
        R__0 = R__0 * R__1;
        goto compute;
```

```
done:
        printf("%d",R__0);
        printf("\n");
}
factorial()
{
        R__2 = 1;
loop:
        R__3 = POP__S();
        if(R__3 == 0) goto ends;
        R__2 = R__2 * R__3;
        R__3 = R__3 - 1;
        PUSH__S(R__3);
        goto loop;
ends:
        PUSH__S(R__2);
}
```

The file "cmm.c"

```
/* C-- version 2.0, June 2, 2005 */
#include <stdio.h>
/* stack element type */
typedef int ITYPE;
typedef float FTYPE;
typedef union u_type { ITYPE ival; FTYPE fval;} S__TYPE;
S__TYPE *STACK__S; /* stack */
ITYPE SP__S; /* stack pointer */
/* integer registers */
ITYPE R__0,R__1,R__2,R__3,R__4,R__5,R__6,R__7,R__8,R__9;
FTYPE F__0,F__1,F__2,F__3,F__4,F__5,F__6,F__7,F__8,F__9;

/* initial stack */
void INIT__S(void)
{
  STACK__S = (S__TYPE *) malloc(sizeof(S__TYPE) * (MAX__S+1));
  SP__S = 0;
}
```

```
/* return top of stack pointer */
ITYPE TOP__S(void)
{
  return(SP__S);
}

/* returns the int value at stack pointer + i */
ITYPE VAL__S(i)
ITYPE i;
{
  return(STACK__S[SP__S+i].ival);
}

/* returns the float value at stack pointer + i */
FTYPE FVAL__S(i)
ITYPE i;
{
  return(STACK__S[SP__S+i].fval);
}
```

```
/* set new stack pointer to be current stack pointer $+ i$ */
void SETSP__S(i)
ITYPE i;
{
  SP__S += i;
}

/* set the int value at stack pointer $+ i$ to the int value $k$ */
void SSET__S(i,k)
ITYPE i;
ITYPE k;
{
  STACK__S[SP__S+i].ival = k;
}

/* set the int value at stack pointer $+ i$ to the int value $k$ */
void FSSET__S(i,k)
ITYPE i;
FTYPE k;
```

```
{
  STACK__S[SP__S+i].fval = k;
}

/* push int value k into stack */
void PUSH__S(k)
ITYPE k;
{
  SP__S += 1;
  STACK__S[SP__S].ival = k;
}

/* push float value k into stack */
void FPUSH__S(k)
FTYPE k;
{
  SP__S += 1;
  STACK__S[SP__S].fval = k;
}
```

```
/* pop int value from stack */
ITYPE POP__S(void)
{
  return(STACK__S[SP__S--].ival);
}

/* pop float value from stack */
FTYPE FPOP__S(void)
{
  return(STACK__S[SP__S--].fval);
}
```