計算機言語講義資料 第5回
C言語

大阪大学　大学院情報科学研究科
2015年
長谷川亨
t-hasegawa@ist.osaka-u.ac.jp

Small C言語

C言語のサブセットである**Small C言語**を定義して、**仮想計算機**の機械語への変換を学ぶことで、プログラムの意味を理解する

Small C言語のC言語との差異

- 基本データ型は、void型とint型のみ
- 構造型は、一次元配列型とポインタ型のみ
- 記憶クラス指定子(static, externなど)、型修飾子(constなど)はない
- 宣言において、初期化はできない
- 関数定義において返り値は、void、int、int *****
- 一次元配列の要素の型は、intまたはint *****
- 文は、複文、式文、if文、while文、break文、return文のみ

Small C言語のプログラム例

```
int i, n;
int f(int i) {
 if (i>0){return i*f(i-1); }
 else {return n+1;}
}
int main (void) {int x;
 i=5; n=3; x=f(n);
 printf ("%d \ n", i+x);
}
```

3

仮想計算機

仮想計算機

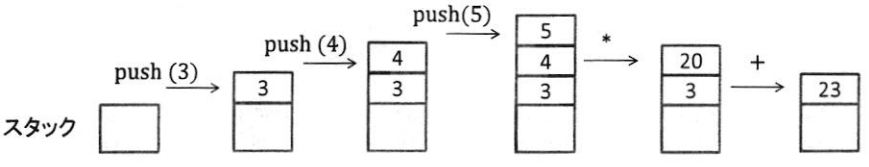
- Small C言語の中間コード、すなわち仮想計算機の機械語を実行する仮想的なスタック計算機

スタック計算機

- メモリがスタックの形式になっている計算モデルを
  - 実行時スタック(テキスト4.4.4節参照)をメモリとして持つ
  - スタックのトップを演算レジスタのように用いて計算

機械語への変換例

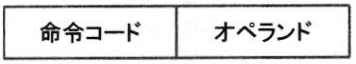
- 逆ポーランド記法(後置記法)による式　3 4 5 * +
  - push (3) push (4) push(5) * +



4

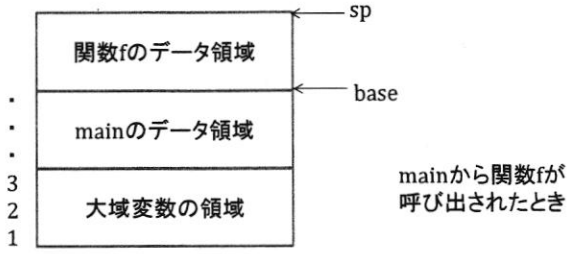
機械語命令

機械語命令　SC-コード命令と呼ぶ
- 命令コード オペランド の形式



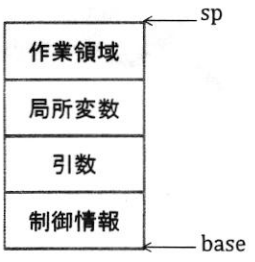
仮想計算機のメモリ領域
- メモリ:機械語のプログラムを格納するためのメモリ
- スタック:実行時スタック

スタック領域のレイアウト



関数のデータ領域とレジスタ

- 関数のデータ領域
  - 制御情報領域
    - 戻り番地(RA)と動的リンク(DL)
      - DL　この関数を呼び出した関数のデータ領域の起点(base)のスタック番地を指すポインタ
  - 引数領域　仮引数に割り当て
  - 局所変数領域　局所変数に割り当て
  - 作業領域
- 3つのレジスタ
  - pc (プログラムカウンタ)
  - base (ベースポインタ)
  - sp (スタックポインタ)



SCコード命令のインタプリタ

- スタック　一次元配列　stack[]　　例) int stack[10000];
  - stack[1]　スタックの底のセル
- レジスタ pc　　int型の変数　　例) int sp;
- レジスタ base　int型の変数　　例) int base;
- レジスタ sp　　int型の変数　　例) int sp;

```
int main(void) {
 int code[2000], operand[2000], stack[10000], sp, base, pc, op, a;
 read; /* SCコードプログラムを読み、code[]とoperand[]に格納 する*/
 base=1; pc=1;
 op=code[1]; a=operand[1];
 while (op!=HLT) {
  pc=pc+1;
  execute; /*命令 op aを実行*/
  op=code[pc]; a=operand[pc];
 }
}
```

7

SC-コード命令

• LDC a	load constant	sp++; stack[sp]=a;
• LLD a	load local	sp++; stack[sp]=stack[base+a];
• LGD a	load global	sp++; stack[sp]=stack[a];
• LLA a	load local address	sp++;stack[sp]=base+a;
• STL a	store local	stack[base+a]=stack[sp];sp--;
• STG a	store global	stack[a]=stack[sp];sp--;
• UPJ a	jump	pc=a;
• FJP a	jump if false	if(stack[sp]==0)pc=a;sp--;

8

SC-コード命令

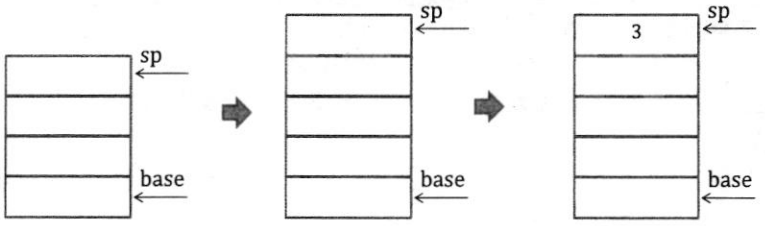
• ENT a	enter block	sp=base+a;
• MST	mark stack	stack[sp+1]=base;sp=sp+2;
• CUP a	call user procedure	stack[sp-a]=pc; base=sp-a-2; pc=stack[base];

• RET a	return from function	sp=base-a;
		pc=stack[base+2];
		base=stack[base+1];

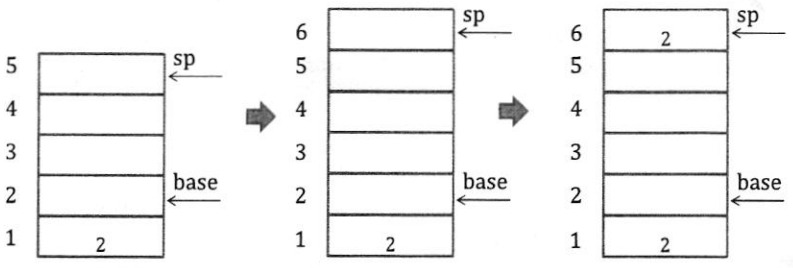
• BOP a	binary operation	sp--;stack[sp]= (stack[sp] a stack[sp+1]);

SC-コード命令の例

LDC 3 load constant sp++; stack[sp]=3;

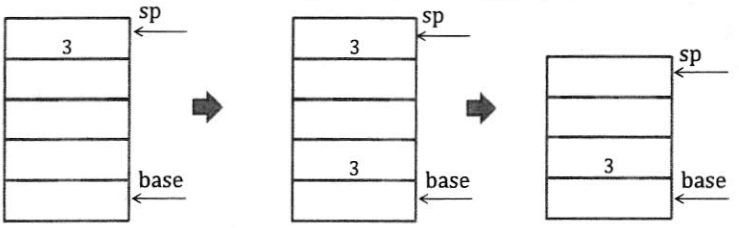


LGD 1 load global sp++; stack[sp]=stack[1];

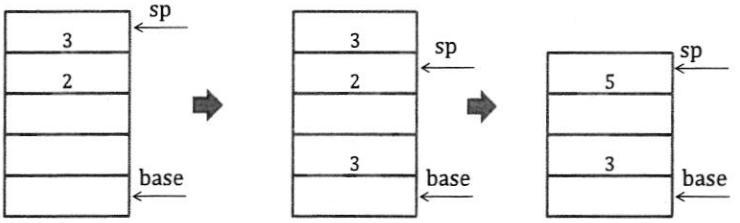


SC-コード命令の例

STL 1 store local　　stack[base+1]=stack[sp];sp--;



BOP + binary operation　　sp--;stack[sp]=(stack[sp] + stack[sp+1]);



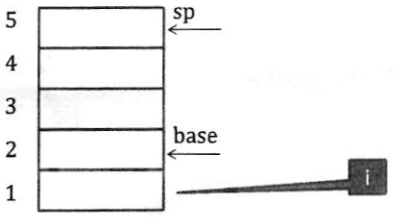
11

問題

演習【11】 整数型の大域変数iを宣言したとき、i=3+4*5; の演算から生成される機械語列(SC-コード命令列)を求めよ。

ここで、大域変数iは、stack[1]に割り当てられているとする。

(ヒント)式3+4*5を後置記法(逆ポーランド記法)に変換する。

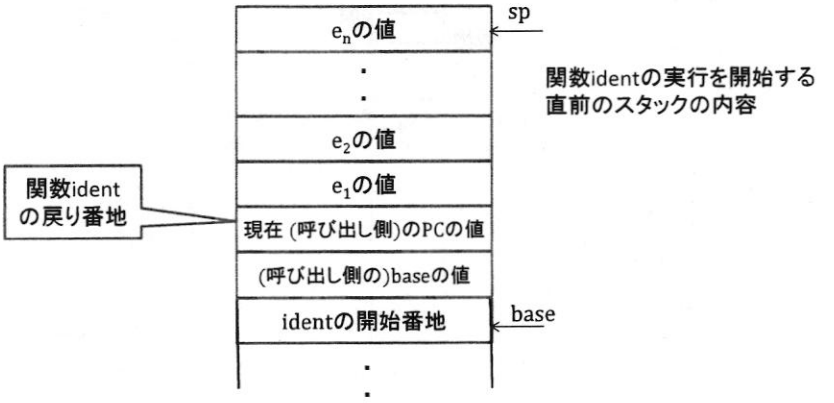


12

関数呼び出しのSC-コード

例　ident (e_1, e_2, ・・・e_n)

- ident は関数名、e_i ($1 \leq i \leq n$)は実引数値



13

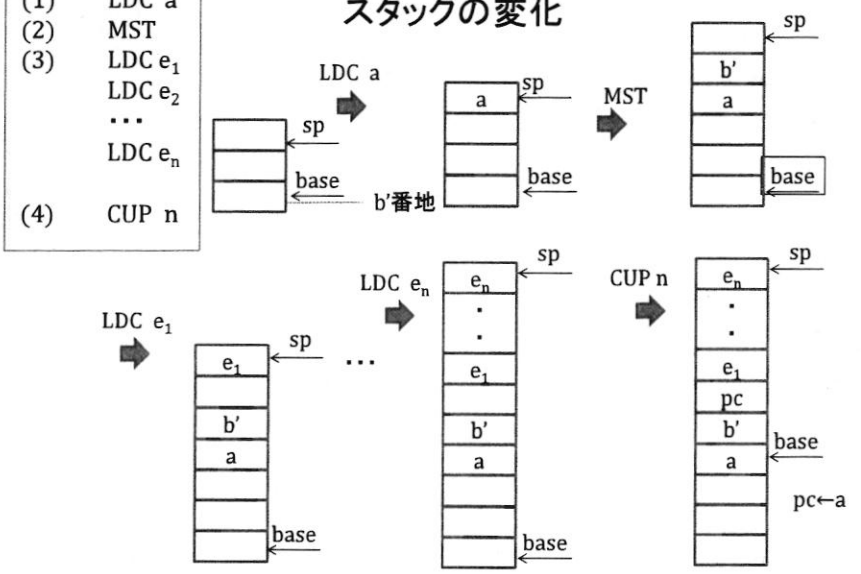
関数identの関数呼び出しから生成されるSC-コード命令

(1)　LDC a　/* aは関数identの実行開始番地 */
(2)　MST
(3)　LDC e_1　/* e_iはコンスタントとする */
　　LDC e_2
　　・・・
　　LDC e_n
(4)　CUP n　/* nは引数の個数を示すコンスタント */

14

スタックの変化

(1) LDC a
(2) MST
(3) LDC e_1
LDC e_2
・・・
LDC e_n
(4) CUP n



aは関数identの実行開始番地　b'は呼び出し側のbaseの値

15

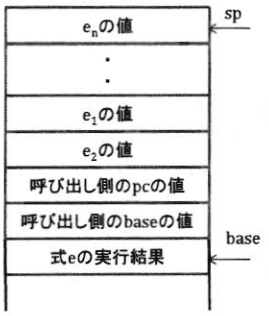
関数の実行終了後の戻り

return e; の実行 /* Small C言語 */

- (1)式eを実行する
- (2)式eの実行結果をstack[base]に格納する
  - stack[base]は、呼び出された関数の実行開始番地と格納と、戻り値の格納の両方に用いられる
- (3)この関数を呼び出した関数に戻る
  - SC-コード命令 RET a

SC-コード命令列 (戻り値がある場合)

STL 0
RET 0



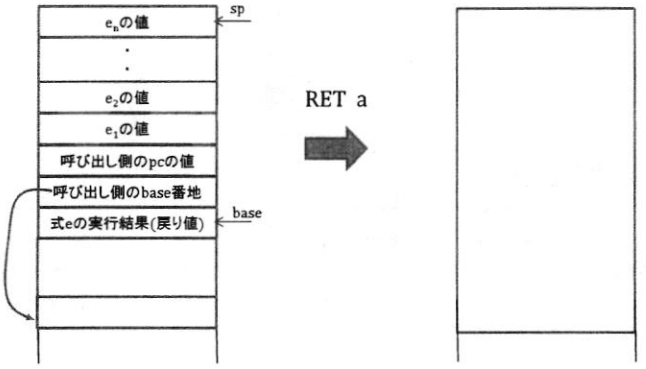
16

問題

【演習12】 RET aにおいて、関数に戻り値が無い場合、a=1 で、戻り値がある場合、a=0である。

戻り値がある場合に、RET aを実行した後のスタックの内容を示せ。

参考RET a　return from function　sp=base-a;pc=stack[base+2];
base=stack[base+1];



関数定義とその呼び出し 例 1

Small Cプログラム

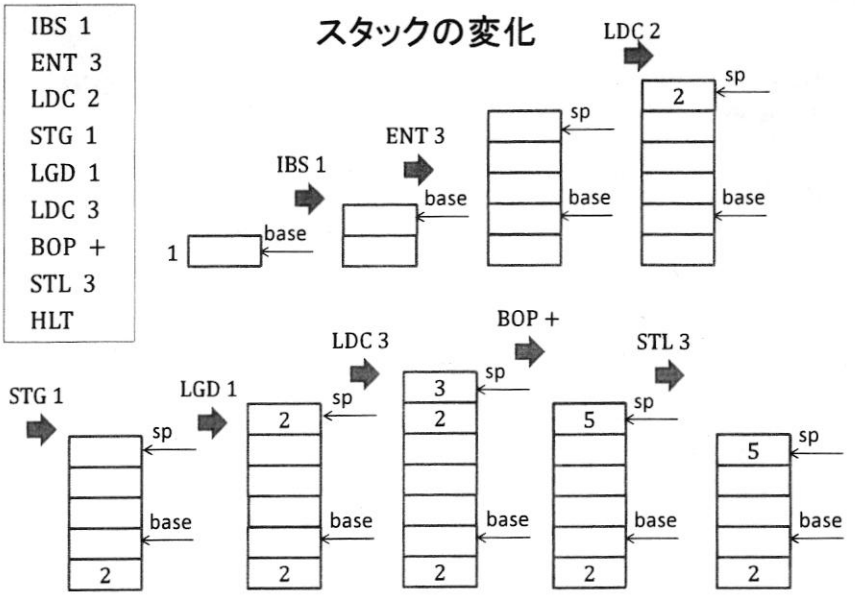
```
int n;
int main(void) {
 int x;
 n=2; x=n+3;
}
```

SC-コードプログラム

```
IBS 1        大域変数の確保
ENT 3        局所変数の確保
LDC 2        定数2をロード
STG 1        大域変数nに格納
LGD 1        nの値をロード
LDC 3        定数3をロード
BOP +     スタックのトップと2番目の値を加算
STL 3        局所変数に格納
HLT
```

```
IBS 1
ENT 3
LDC 2
STG 1
LGD 1
LDC 3
BOP +
STL 3
HLT
```

スタックの変化



19

関数定義とその呼び出し 例2

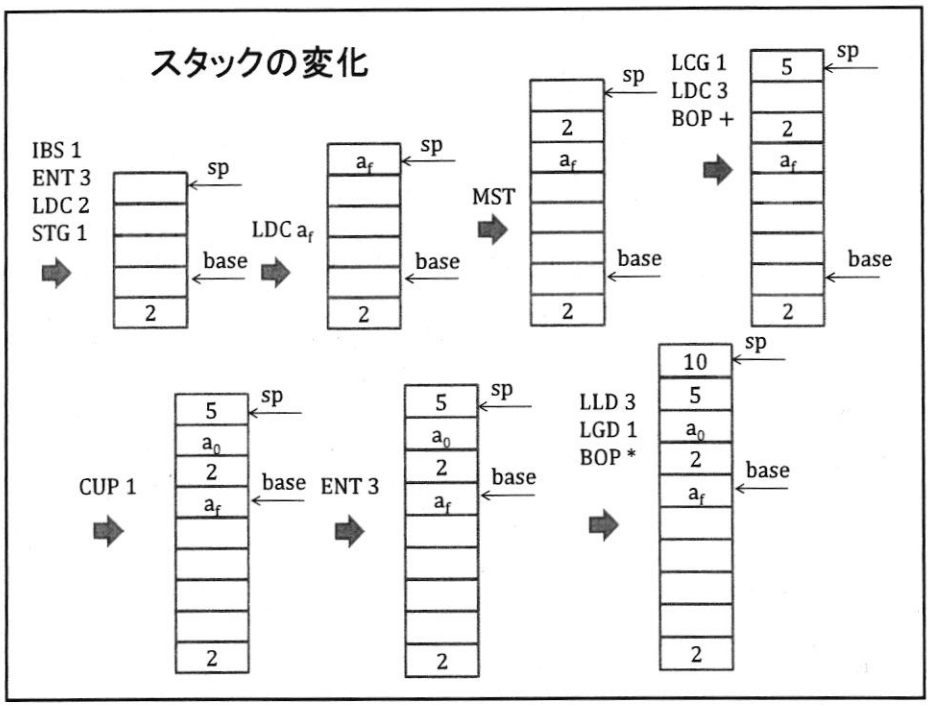
Small Cプログラム

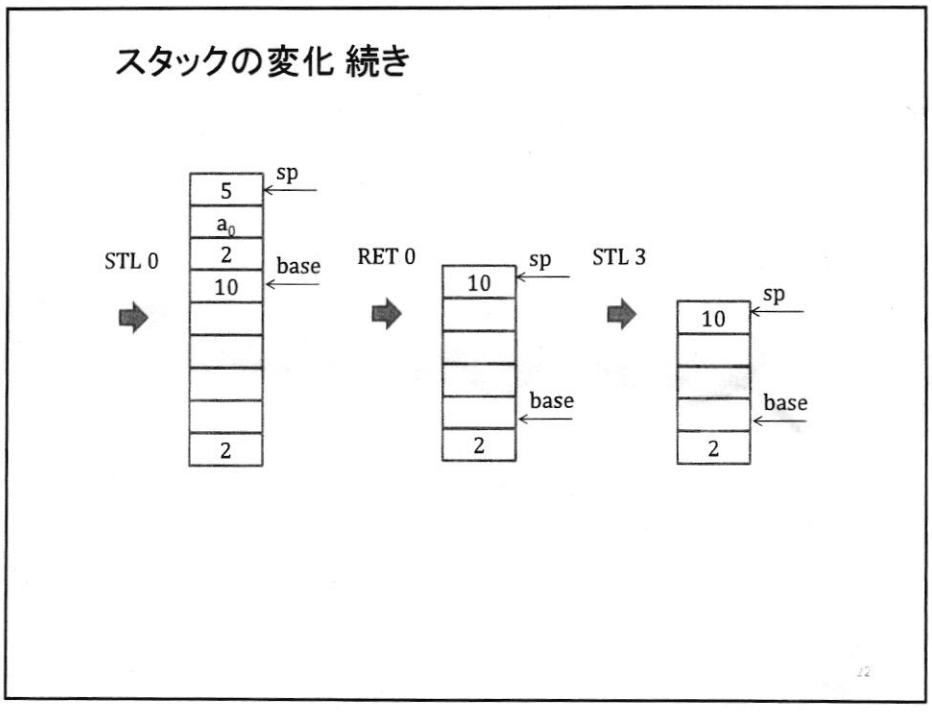
```
int n;
int f(int i) {
  return i*n;
}
int main(void) {
  int x;
  n=2;
  x=f(n+3);
}
```

SC-コードプログラム

```
      IBS  1          af:   ENT 3
am:   ENT  3                LLD 3
      LDC  2                LGD 1
      STG  1                BOP *
      LDC  af               STL 0
      MST                   RET 0
      LGD  1
      LDC  3
      BOP+
      CUP  1
a0:   STL  3
      HLT
```

20


スタックの変化
IBS 1
ENT 3
LDC 2
STG 1
LDC a_f
MST
LCG 1
LDC 3
BOP +
CUP 1
ENT 3
LLD 3
LGD 1
BOP *
sp
base
2
a_f
5
a_0
10


スタックの変化 続き
STL 0
RET 0
STL 3
sp
base
5
a_0
2
10