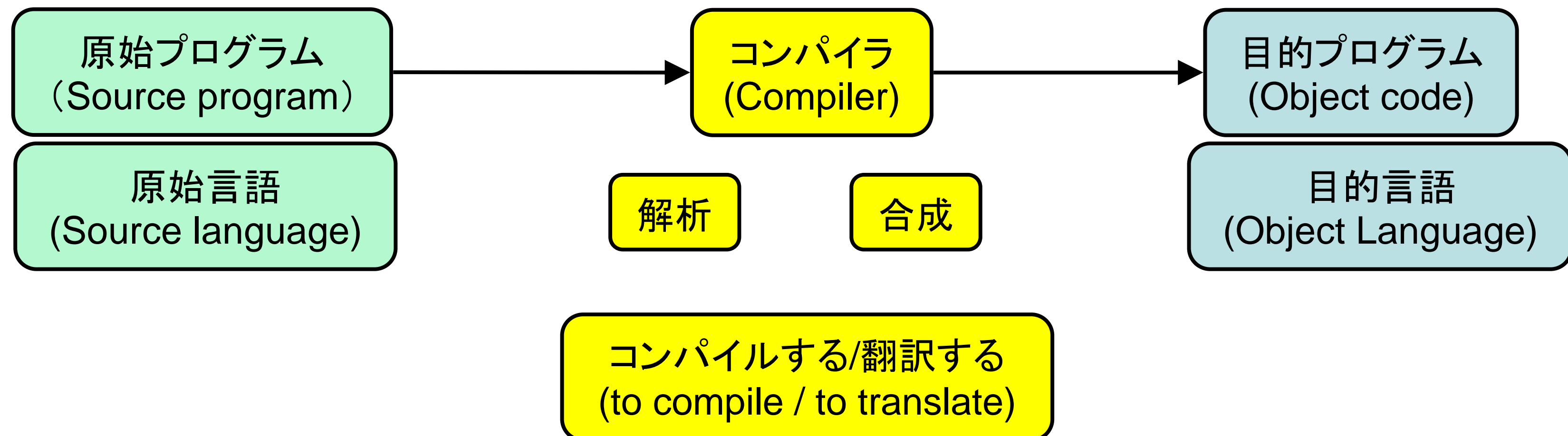
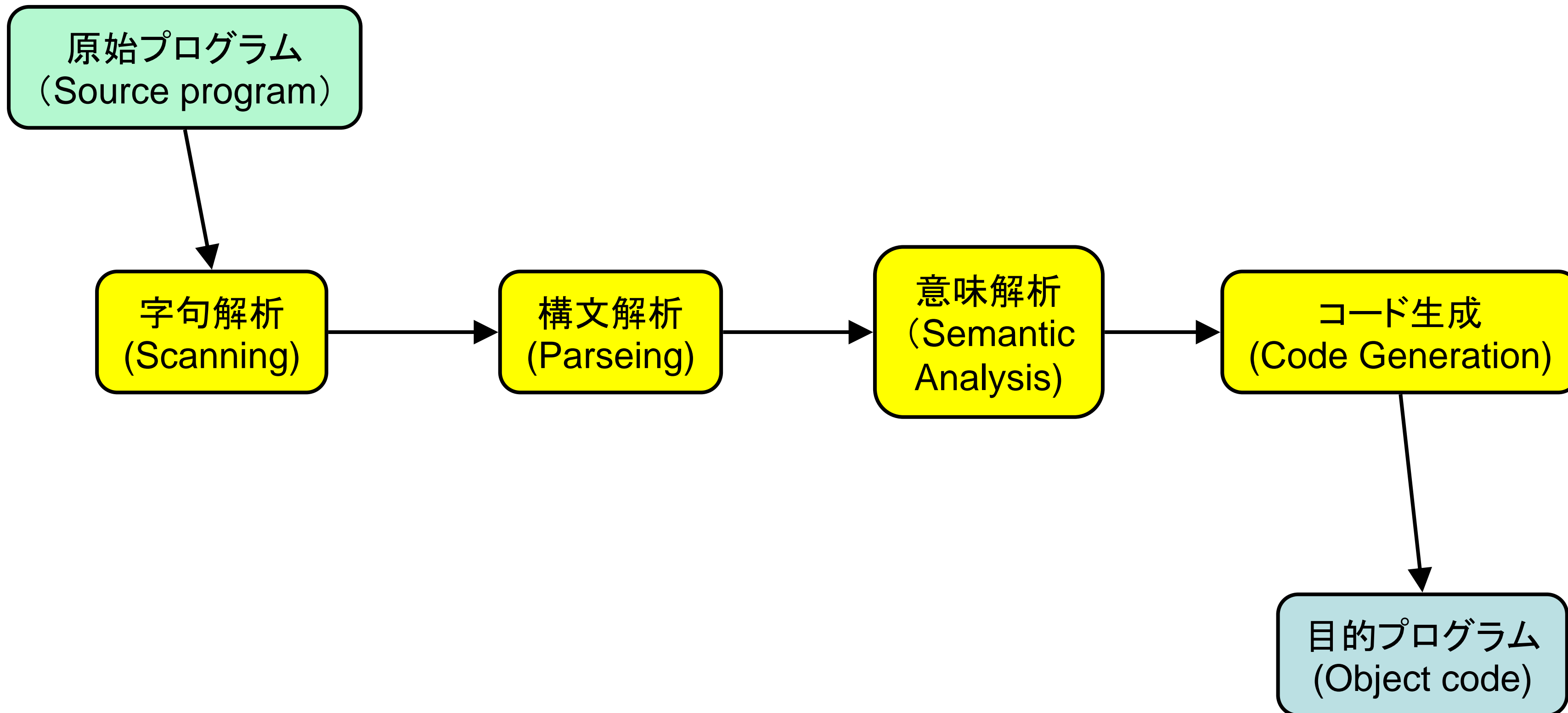
仮想機械(VMの紹介）

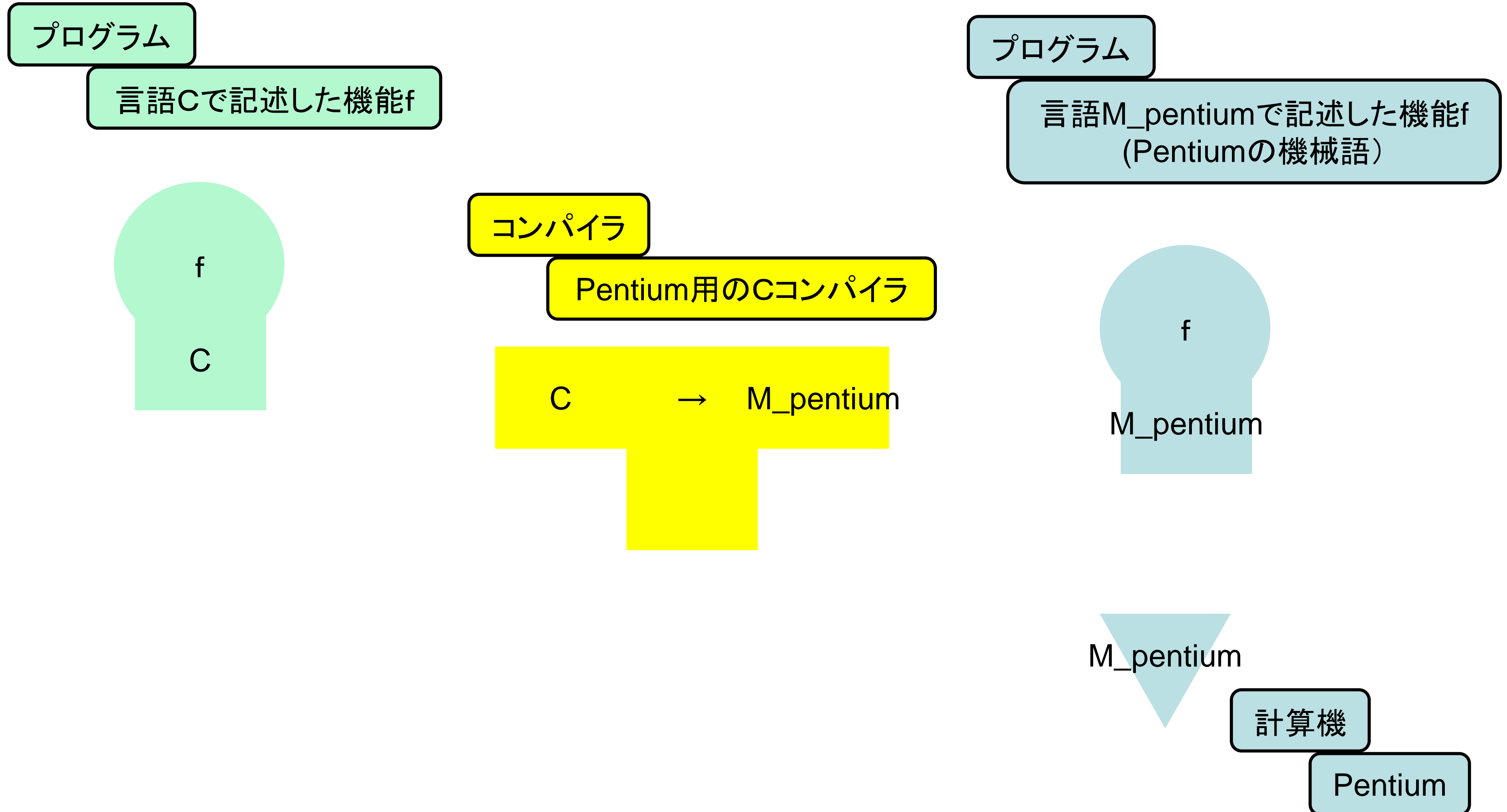
プログラム言語の処理系(コンパイラ)



コンパイラのフェーズ

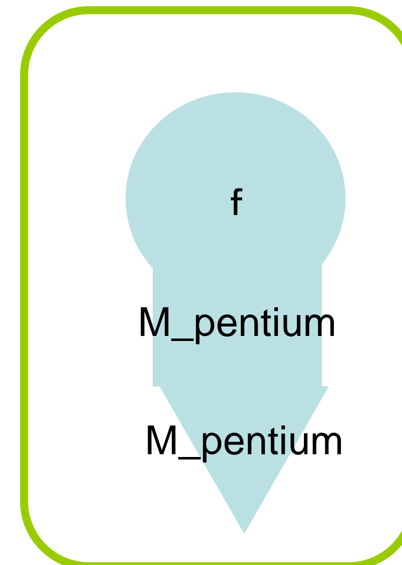


T Diagram



ある機能fの実現(Implementation)

- fを機械語(Pentium)で記述する。
- Pentiumの上で実行。



C言語による言語処理過程

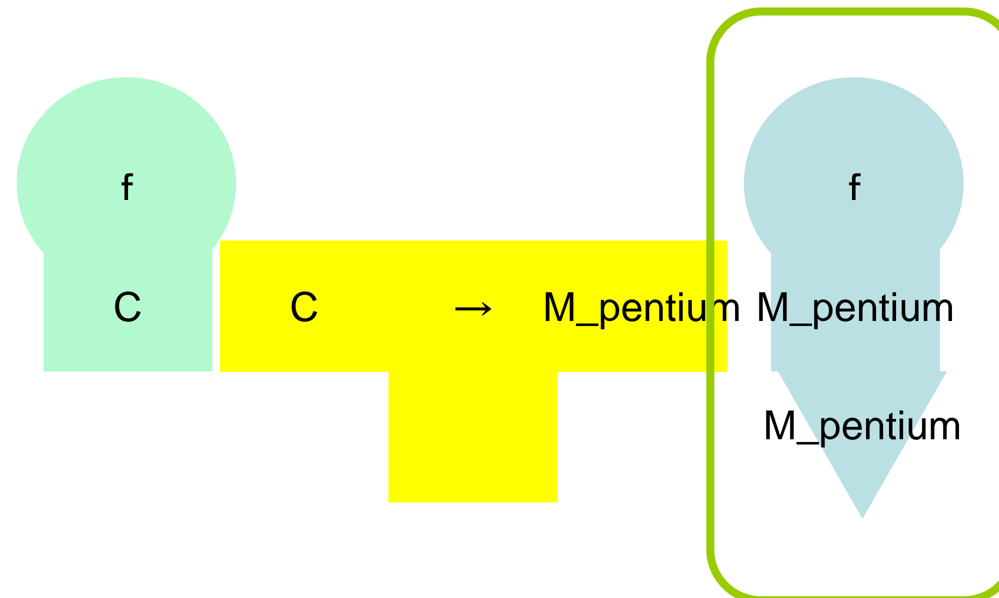
- 機能fの実現

  fをC言語のプログラムとして記述する

  C Compilerでコンパイルする

  機械語(Pentium)によるfに変換された！

  Pentiumの上で実行



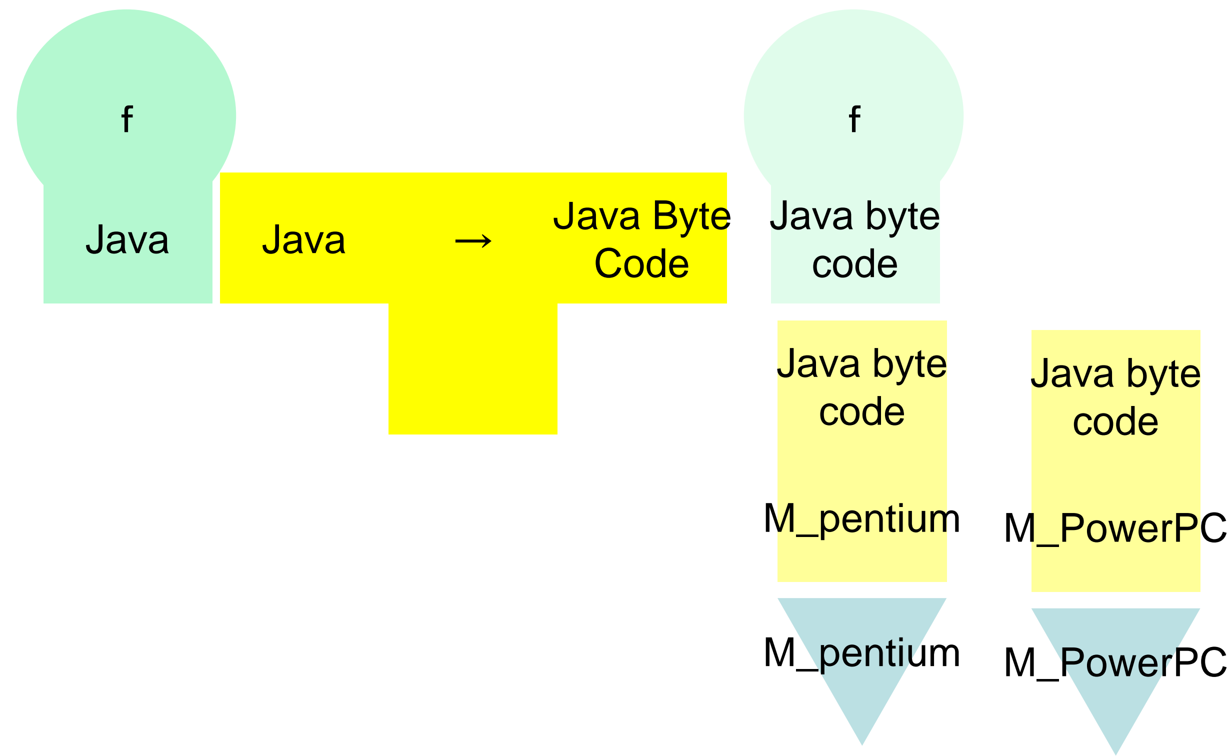
Java言語による言語処理過程

- 機能fの実現

  Javacでコンパイルする

  Javaバイトコードよるfに変換された！(F.class)

  Jvm(Java virtual machine)上で実行



コンパイラ・インタプリタ方式と仮想マシン

- 原始プログラムをその言語に適した仮想的な計算機（仮想機械, Virtual Machine)の機械語に変換する。
  - 現実の計算機の機械語に変換するより容易にコンパイラを作成可能
- VMを用いる利点、欠点
  - ○移植性が高い(マシン独立)
  - ○目的コードが小さい
  - ×実行速度が遅い
    - → JIT (Just In Time)コンパイラ技術など

Java言語の例

Test.java

```
public class Test{
…
}
```

Javaコンパイラ
(javac Test.java)

Test.class

b22b 1700 0303
00b6 5718 b62b
1900 2b57

Java仮想機械
(java Test)

Java言語の例

Test.java

```
public class Test{
  public static void main(String[] arg){
     int i=3;
     int j=4;
     System.out.println(i+j);
  }
```

Test.classの内容(一部)
od –h Test.classで表示

```
…
0001 2804 2949 0056 0021 0004 0005 0000
0000 0002 0001 0006 0007 0001 0008 0000
001d 0001 0001 0000 2a05 00b7 b101 0000
0100 0900 0000 0600 0100 0000 0100 0900
…
```

Test.classを逆アセンブル
javap –c Testで表示

```
0:   iconst_3
1:   istore_1
2:   iconst_4
3:   istore_2
4:   getstatic       #2;
7:   iload_1
8:   iload_2
9:   iadd
10:  invokevirtual       #3;
13:  return
```

演習で作るコンパイラの例

test.hcc

```
Int main()
{    int i j;
     i = 3;
     j = 4;
     putint(i+j);
 }
```

test.hsm

PUSH	0	2
LDC	0	3
STV	0	0
LDC	0	4
STV	0	1
LDV	0	0
LDV	0	1
AD	0	0
WRI	0	0
POP	0	2
HLT	0	0

Test.classを逆アセンブル javap –c Testで表示

```
0:    iconst_3
1:    istore_1
2:    iconst_4
3:    istore_2
4:    getstatic    #2;
7:    iload_1
8:    iload_2
9:    iadd
10:   invokevirtual        #3;
13:   return
```

Hsm (HiStackMachine)の概要(1)

- 演習で用いる仮想機械(スタックマシン）
- 構成

プログラムP

・命令列の置き場

プログラムカウンタ(pc)

・次に実行する命令を指示

スタック(S)

・演算対象（被演算数、演算結果）を置く

・記憶域

スタックポインタ(sp)

・スタックトップを指す

フレームポインタ(fp)

・関数（手続き）のフレームの開始アドレス（後の講義で説明）

Hsm (HiStackMachine)の概要(2)

- 命令セット

(1) ロード・ストア命令

・ロード命令：スタックトップに値を置く。

・ストア命令：記憶域として確保した所に値を保存する。

・記憶域の確保、開放の命令

(2) 演算命令

・算術演算、関係演算。(論理演算はない)

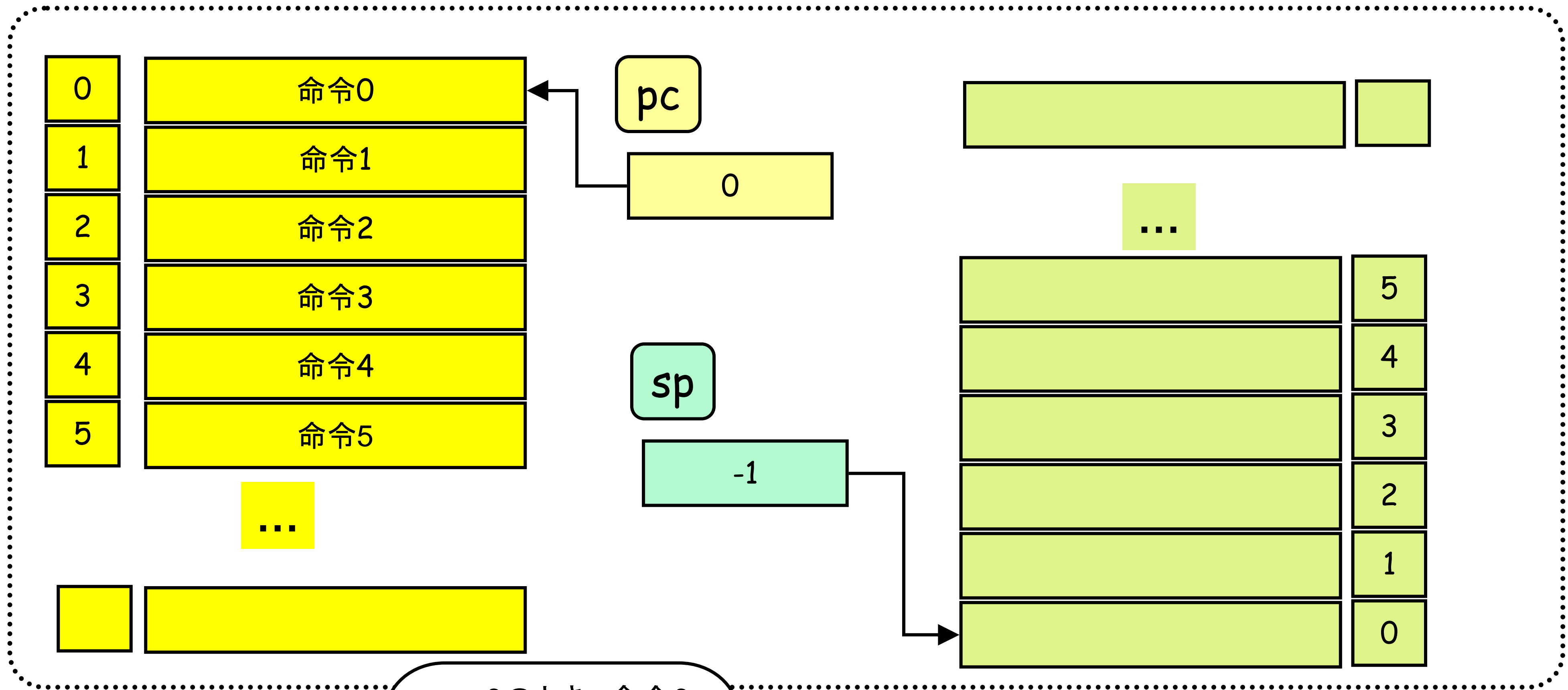
(3) ジャンプ命令、制御命令

・無条件ジャンプ、条件ジャンプ、停止命令

(4) 入出力命令

・入力、出力

hsmの構成図



P

pc = 0のとき、命令0を実行する。
pc = 1のとき、命令1を実行する
…
pc = nのとき命令nを実行する

S

値をロードする場合には、sp←sp+1としてS[sp]に置く。

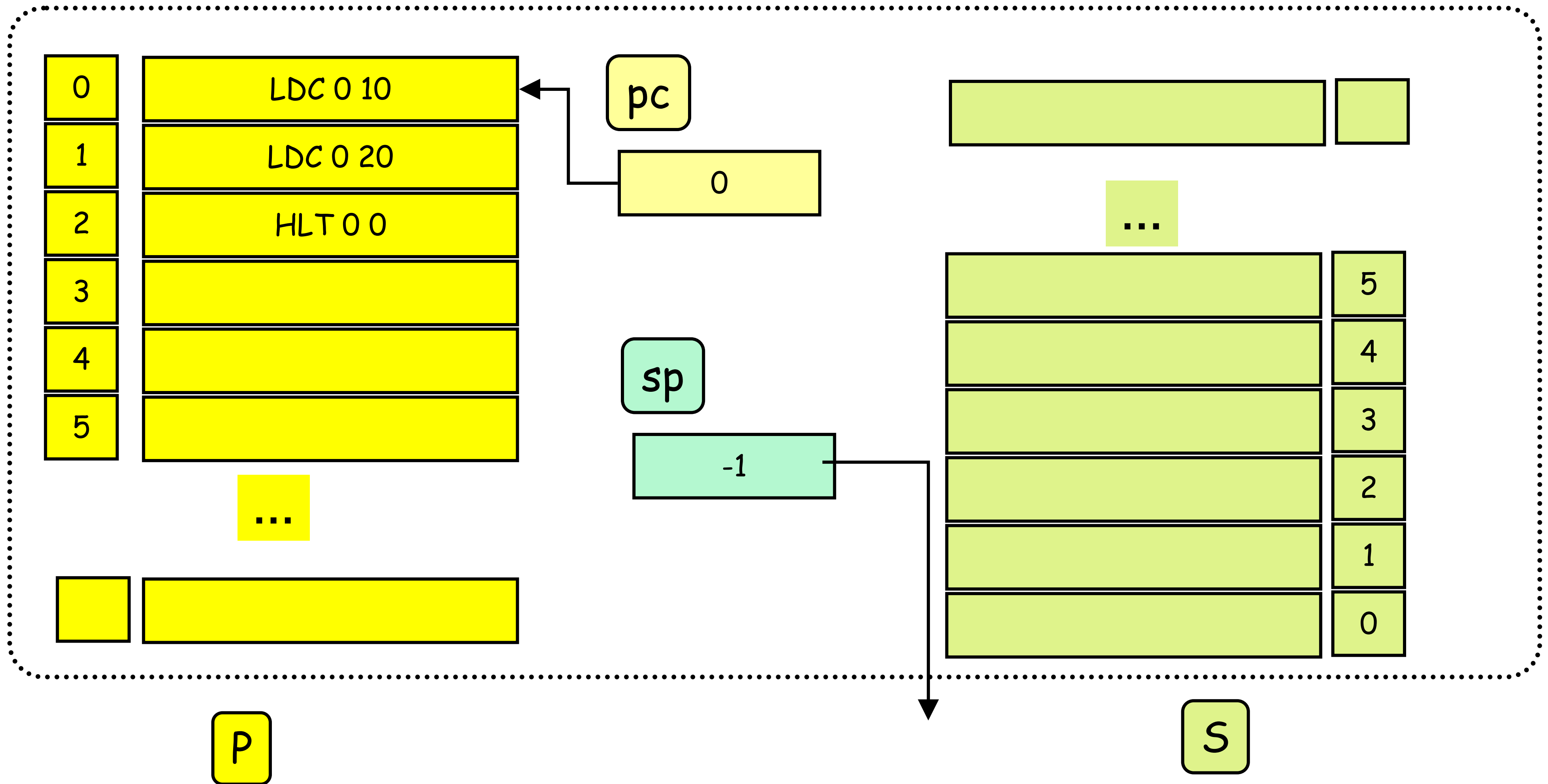
hsmの実行例

```
HLT  0  0      .. プログラムの終了
LDC  0  N      … Nをスタックトップの上に積む。
                 sp++ ; S[sp] ← N ; pc++
```

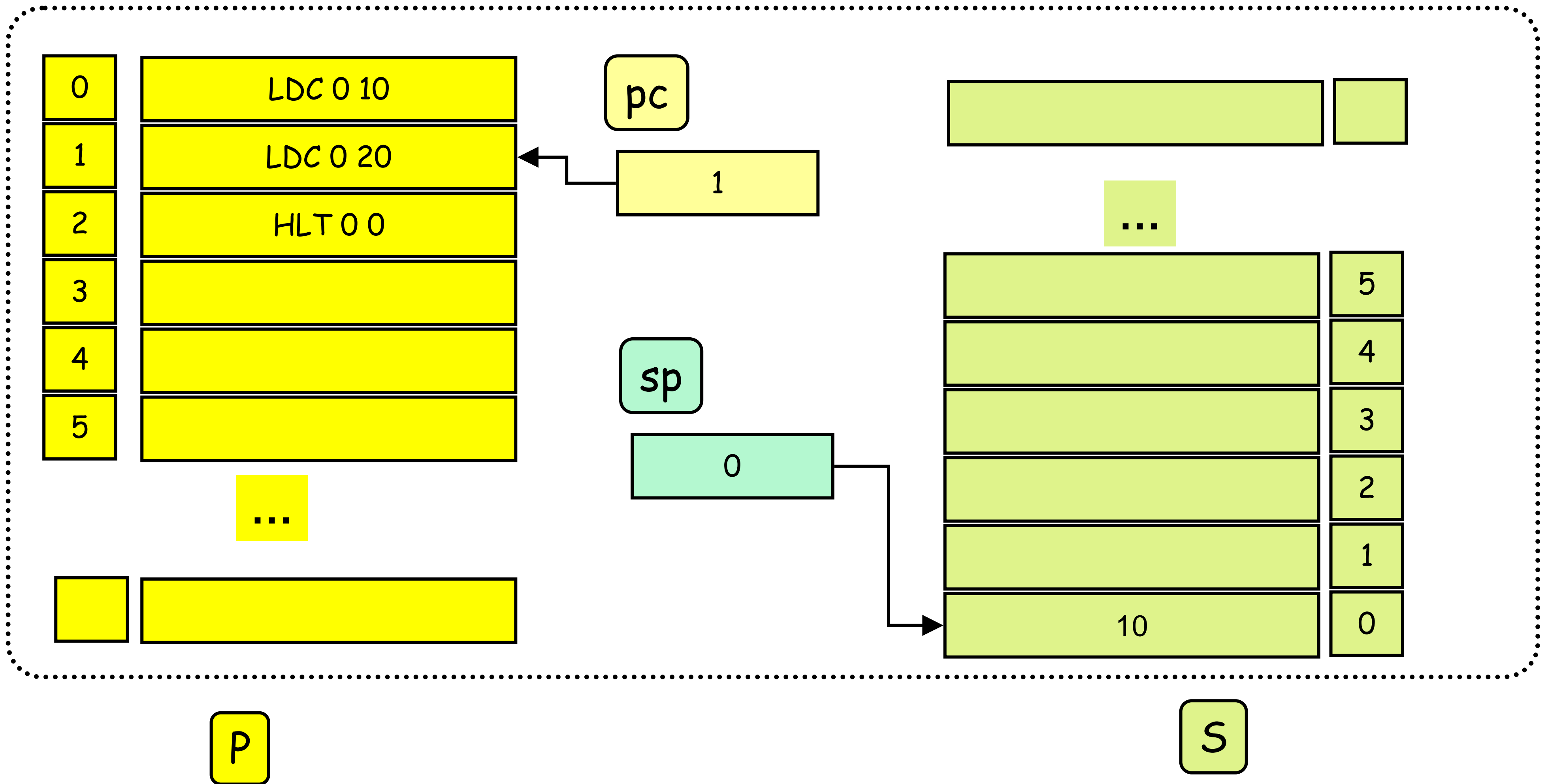
注:
spは、スタックトップ（一番最後に値をロードした場所）を指す解釈と、次に値をロードする場所(一番最後に値をロードした場所の一段上）を指す解釈が可能。

本講義、演習では前者の解釈を採用する。

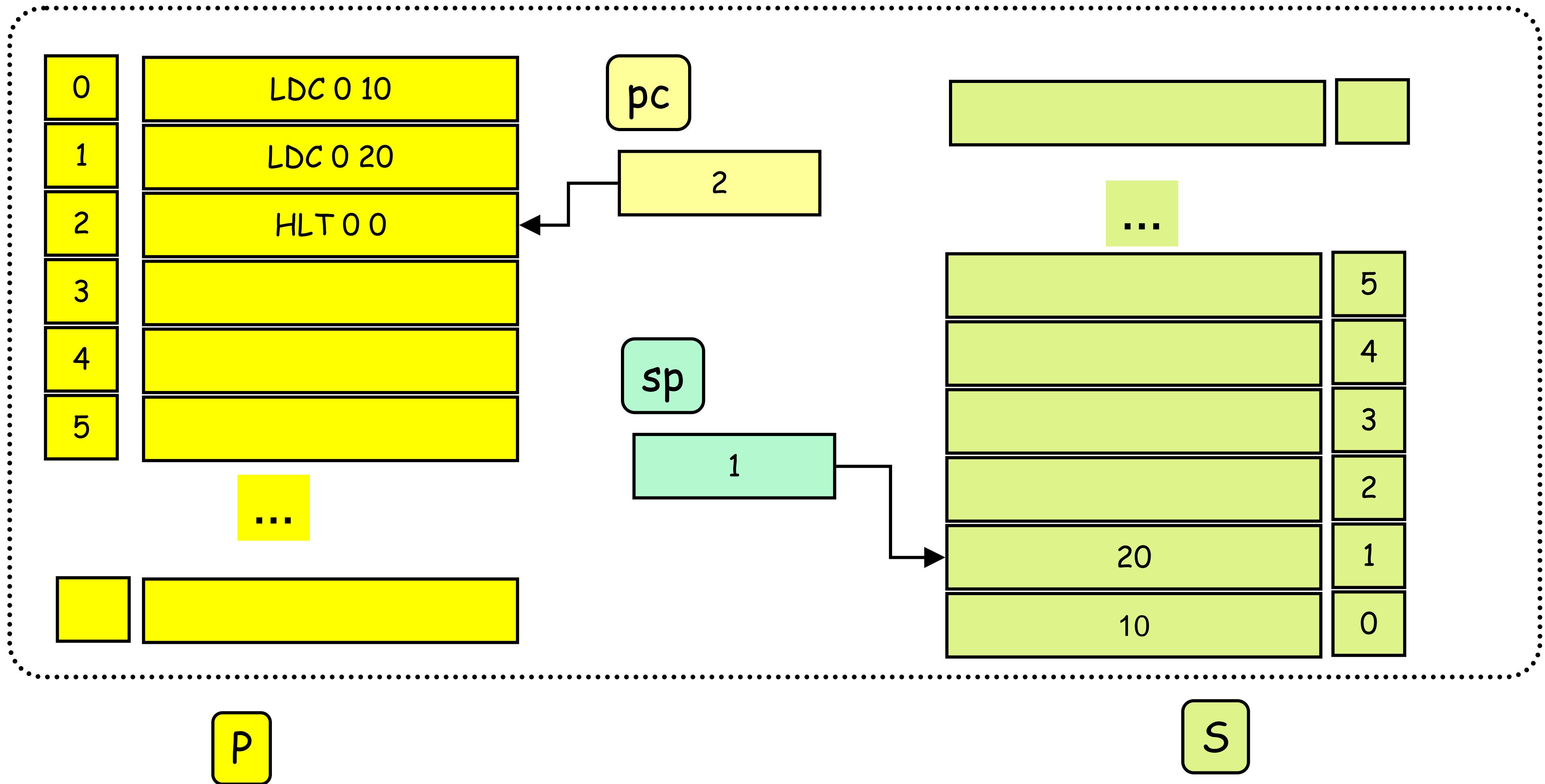
実行例(1)



実行例(2)



実行例(3)



演算命令

SB 0 0　　引き算命令:

　　sp-- ; S[sp] ←S[sp]-S[sp+1]; pc++

NEG 0 0　　符号反転命令:

　　S[sp] ← -S[sp]; pc++;

LE 0 0　　関係演算命令<=

　　sp-- ; if (S[sp] <= S[sp+1]) then S[sp]←1

　　　　else S[sp] ← 0; pc++

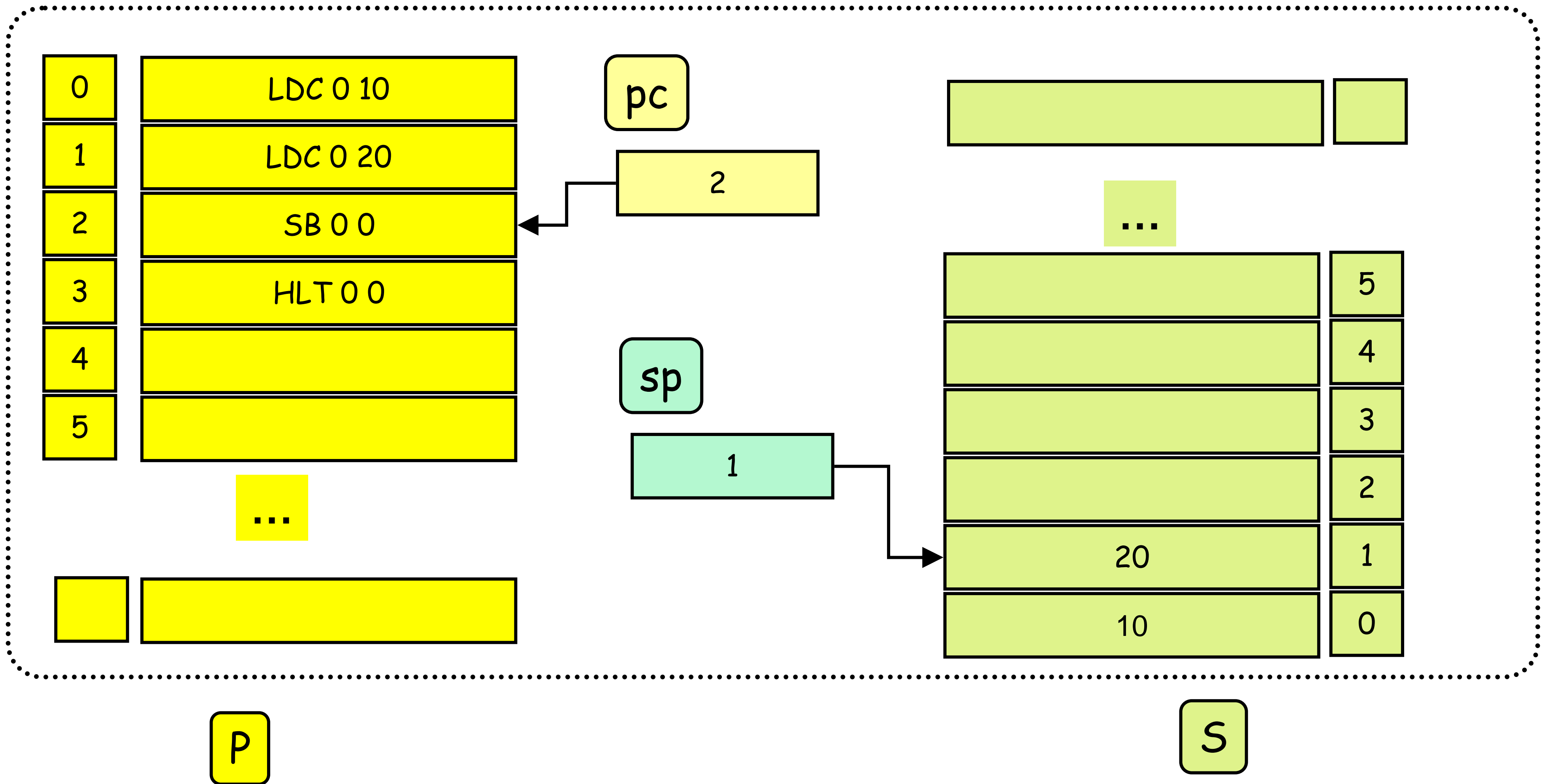
算術演算:

AD, SB,ML,DV, NEG … +, -, *, /, 反転

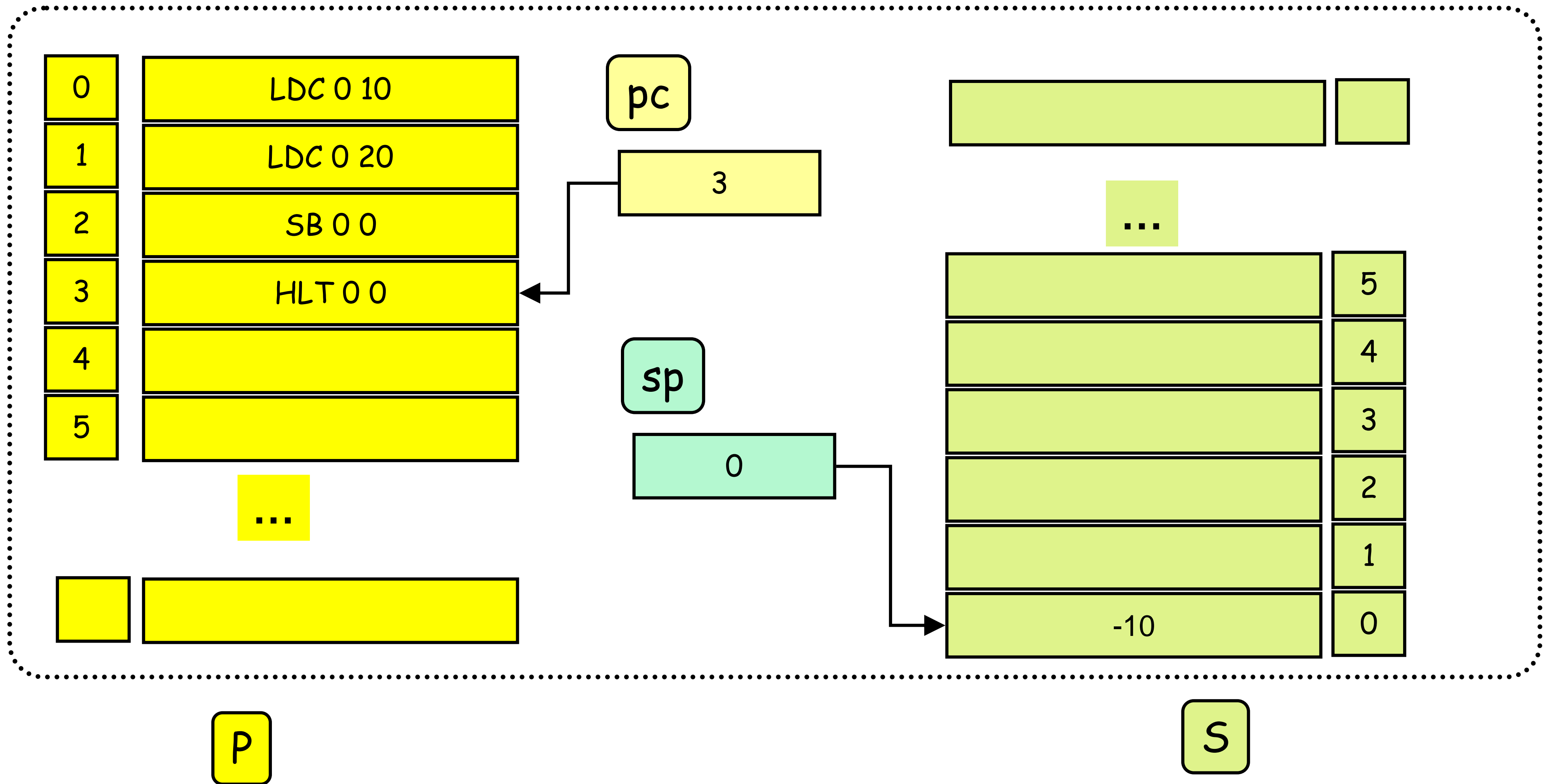
関係演算:

EQ, NEQ, LT,LE,GT,GE … ==, !=, <, <=, >, >=

実行例(4)



実行例(5)



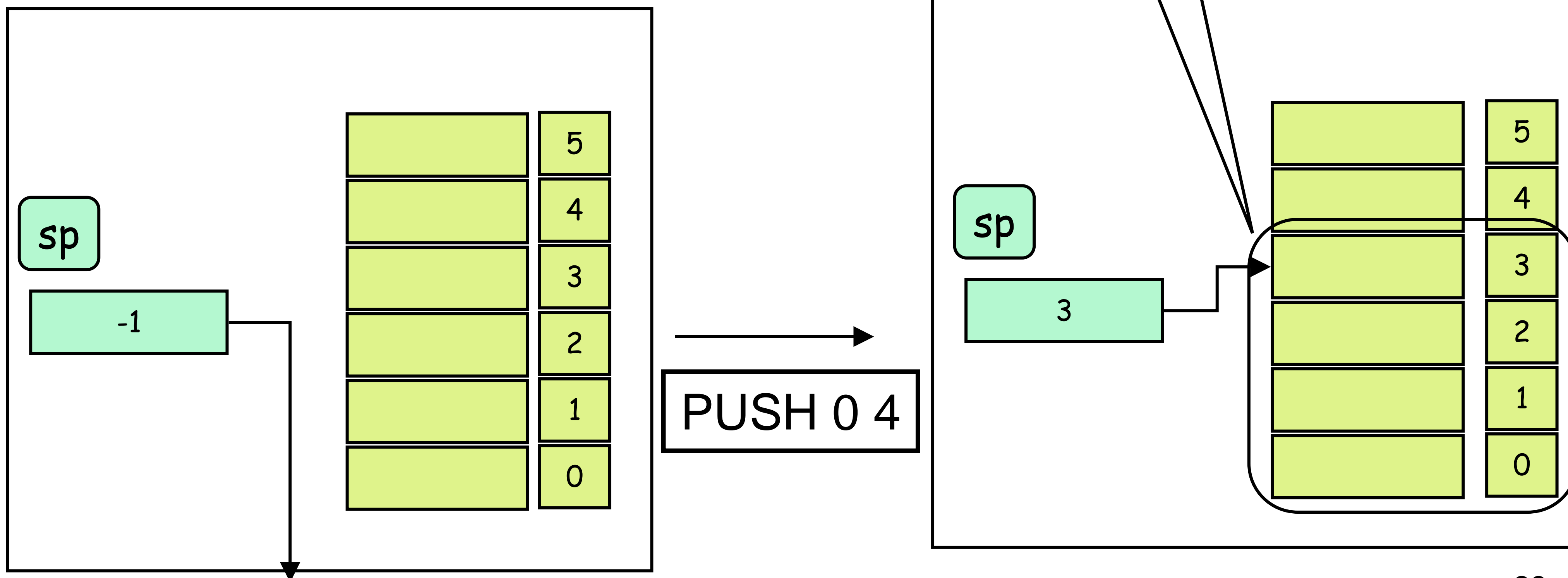
ロード命令、ストア命令(暫定版)

STV 0 N　ストア命令
S[N] ←S[sp]; sp--; pc++
LDV 0 N　ロード命令
sp++; S[sp]←S[N]; pc++
LDC 0 N　即値ロード命令
sp++; S[sp]←N; pc++

Hsmのwebページの説明では、
STV p q
s[base(p)+q]=s[t];t=t-1;pc=pc+1;
となっている。p=0のときは、base(p)=0である。Pが0以外の場合については後の講義で説明する。

メモリの確保と開放

PUSH 0 N　　メモリを確保
　　　　　　　sp←sp+N; pc++
POP 0 N　　メモリを開放
　　　　　　　sp←sp-N; pc++



コンパイラ作成の流れと、プログラムの実行