Systems Software: P-Machine Example #1

Andrew Harn

University of Central Florida

PL/0 Language

- The high level language used for the P-Machine

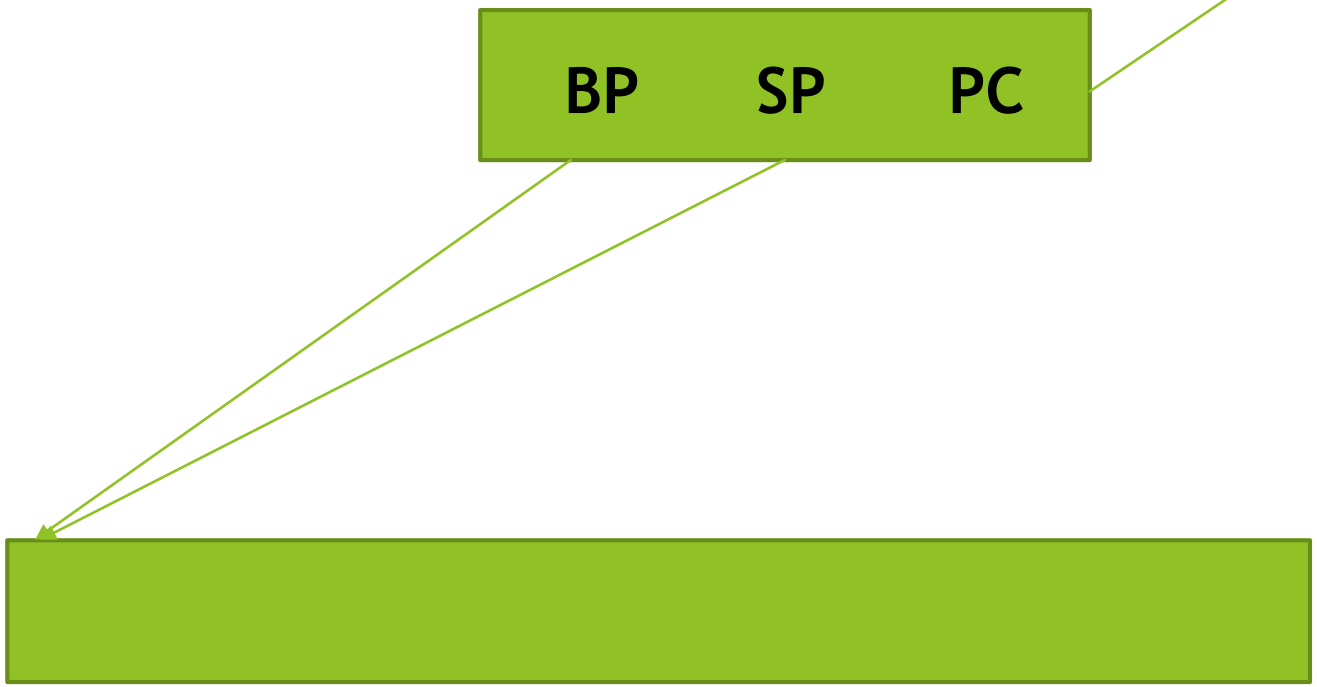
```
const n = 13;                  /* constant declaration */
var i,h;                       /* variable declaration */
procedure sub;                 /* subroutine declaration */
  const k = 7;
  var j,h;
  begin
    j:=n;
    i:=1;
    h:=k;
  end;
begin                          /* main routine starts here */
  i:=3;
  h:=0;
  call sub;
end.
```

The compiled PL/0 language

- The previous code is compiled into the assembly language for the P-Machine
- We will go over the execution of this code while watching the state of the machine

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Execution

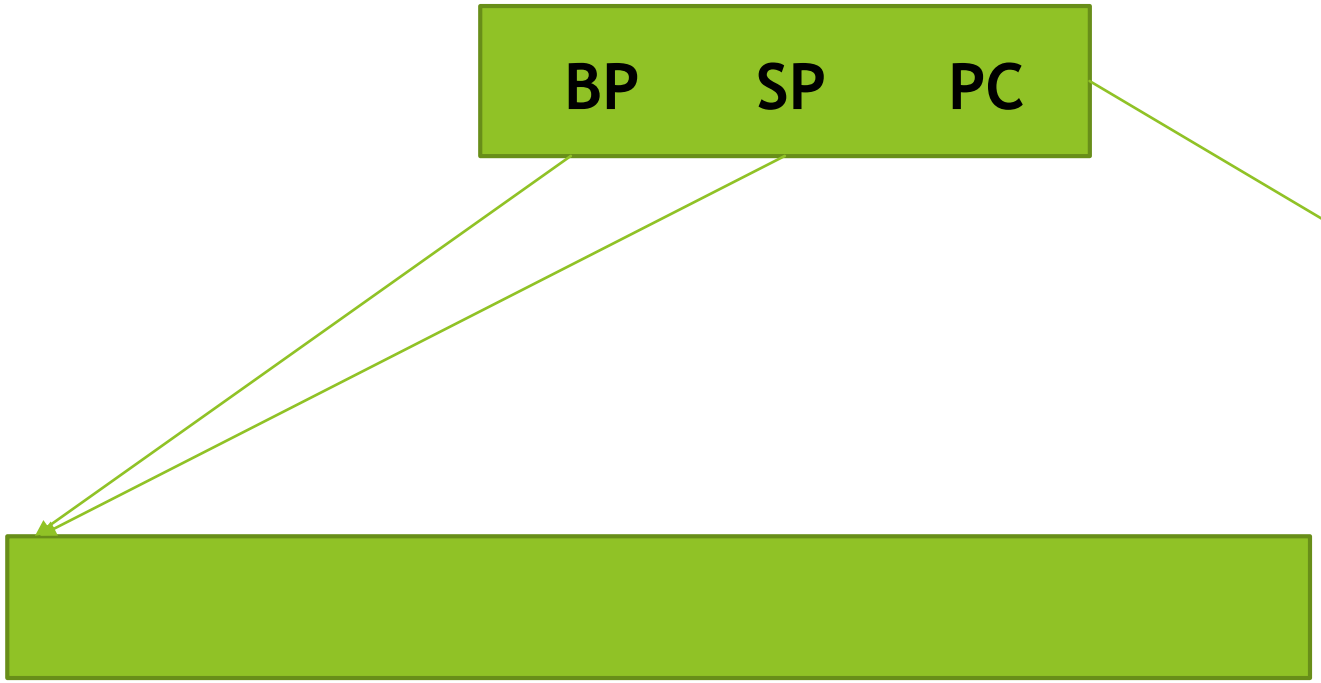


AR Stack

```
0 JMP 0 10
1 JMP 0 2
2 INC 0 5
3 LIT 0 13
4 STO 0 3
5 LIT 0 1
6 STO 1 3
7 LIT 0 7
8 STO 0 4
9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution

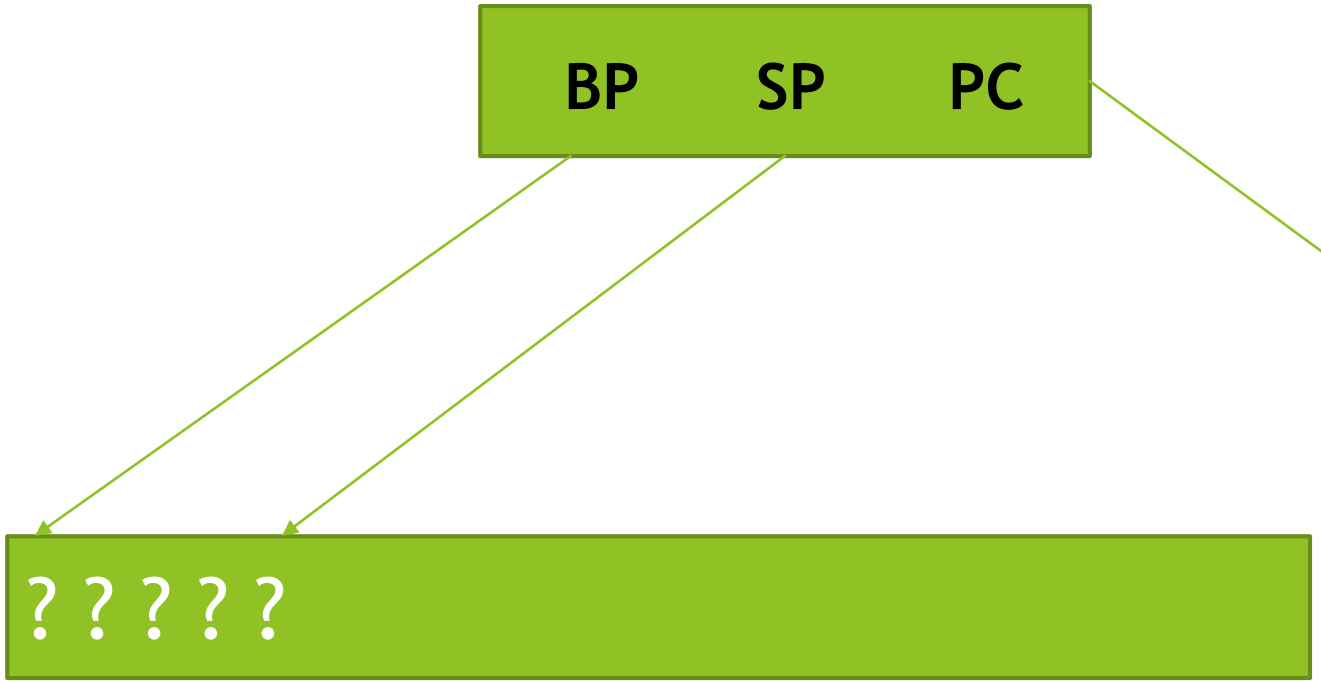


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution

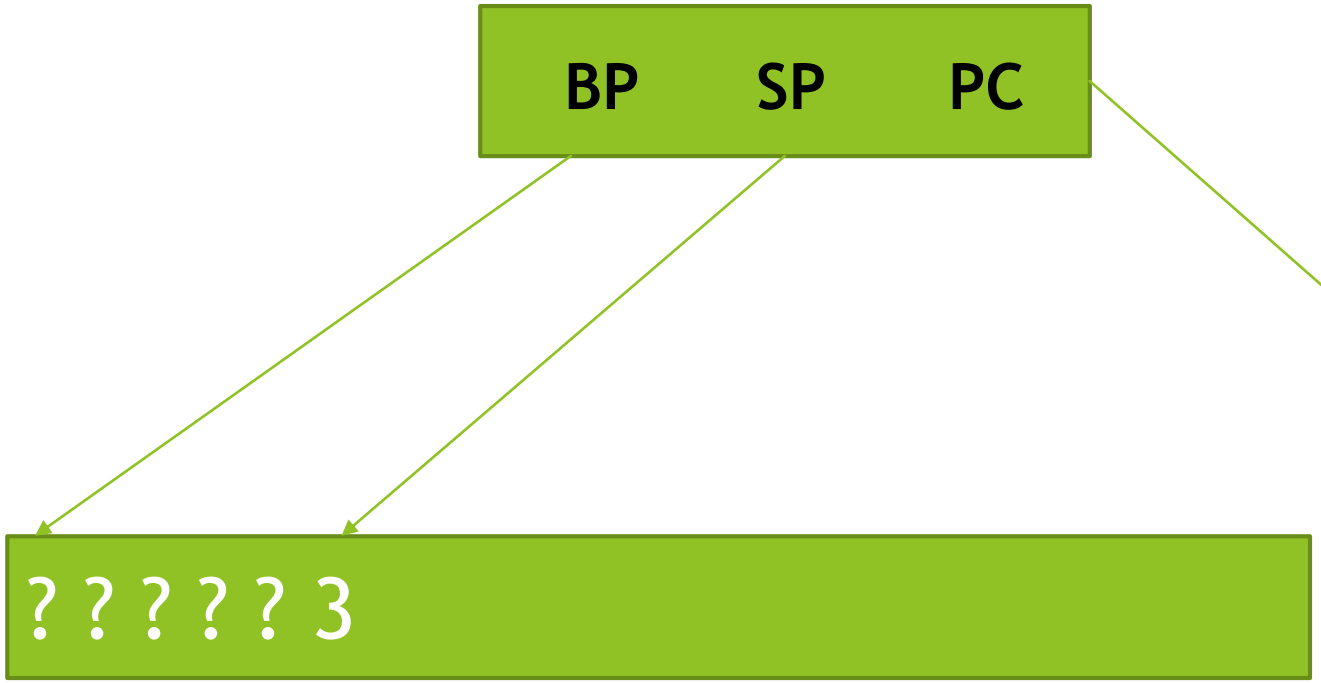


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution

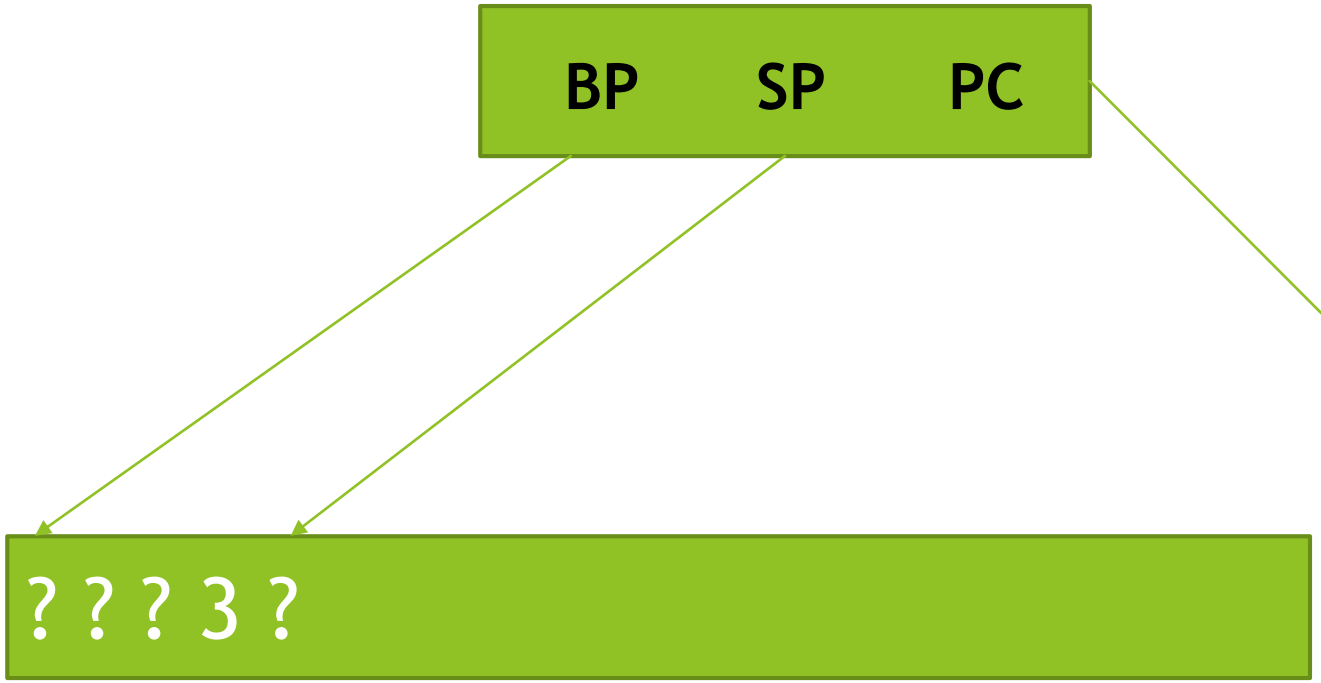


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution

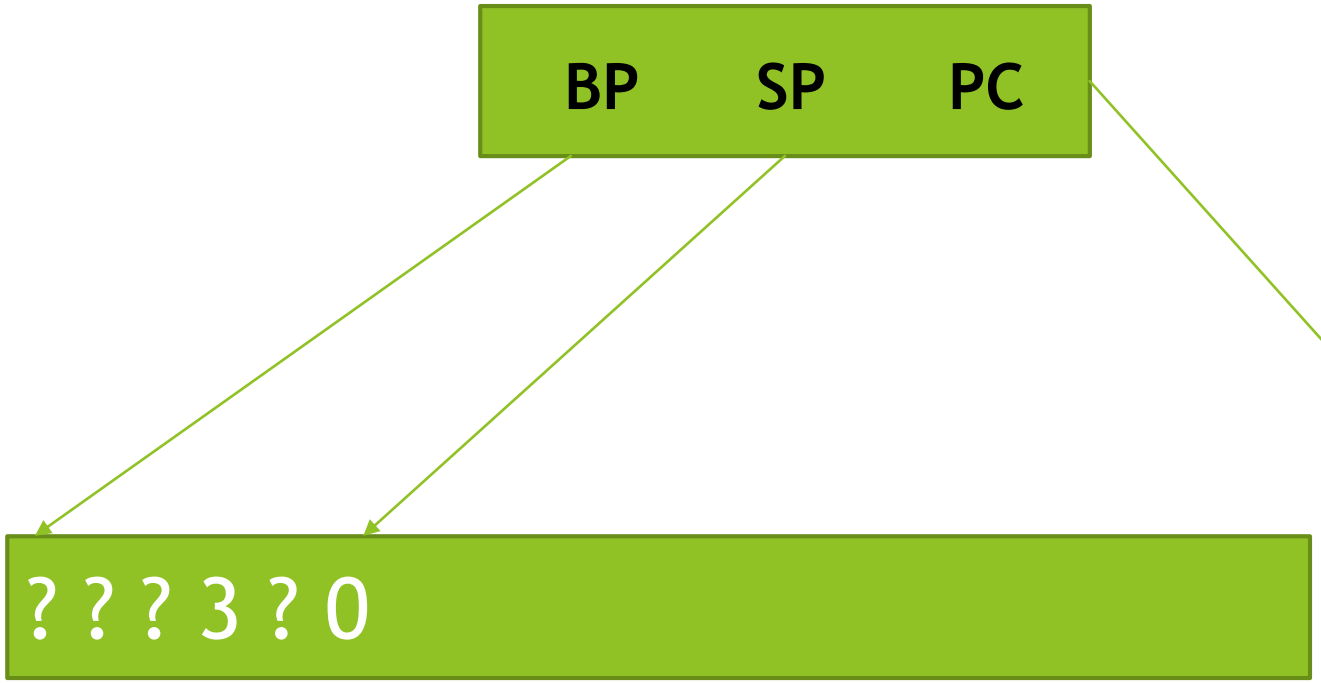


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution

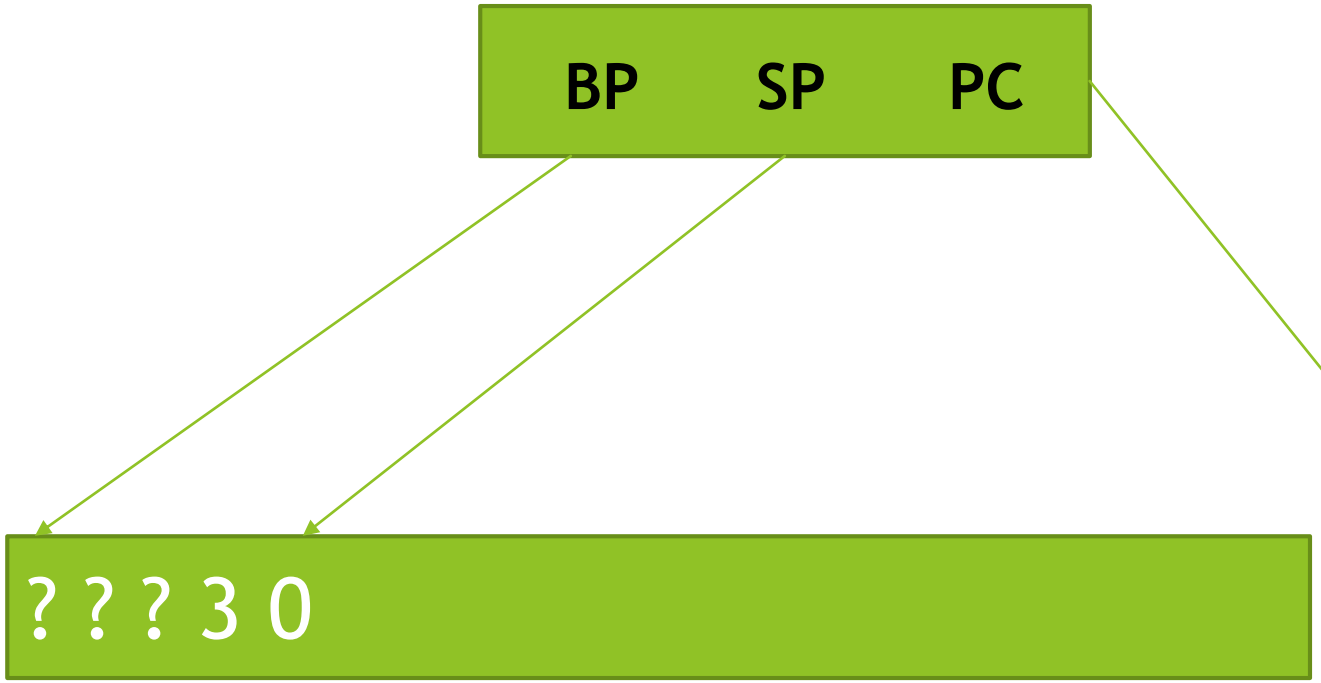


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution

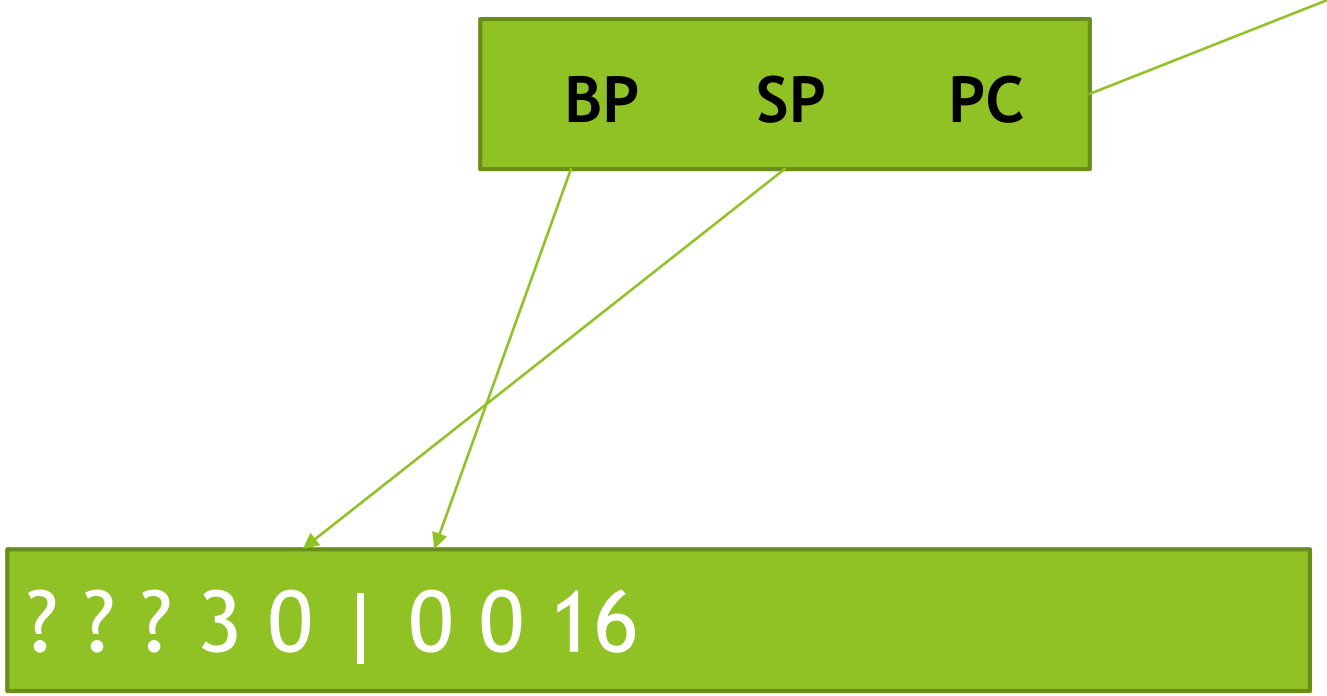


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution



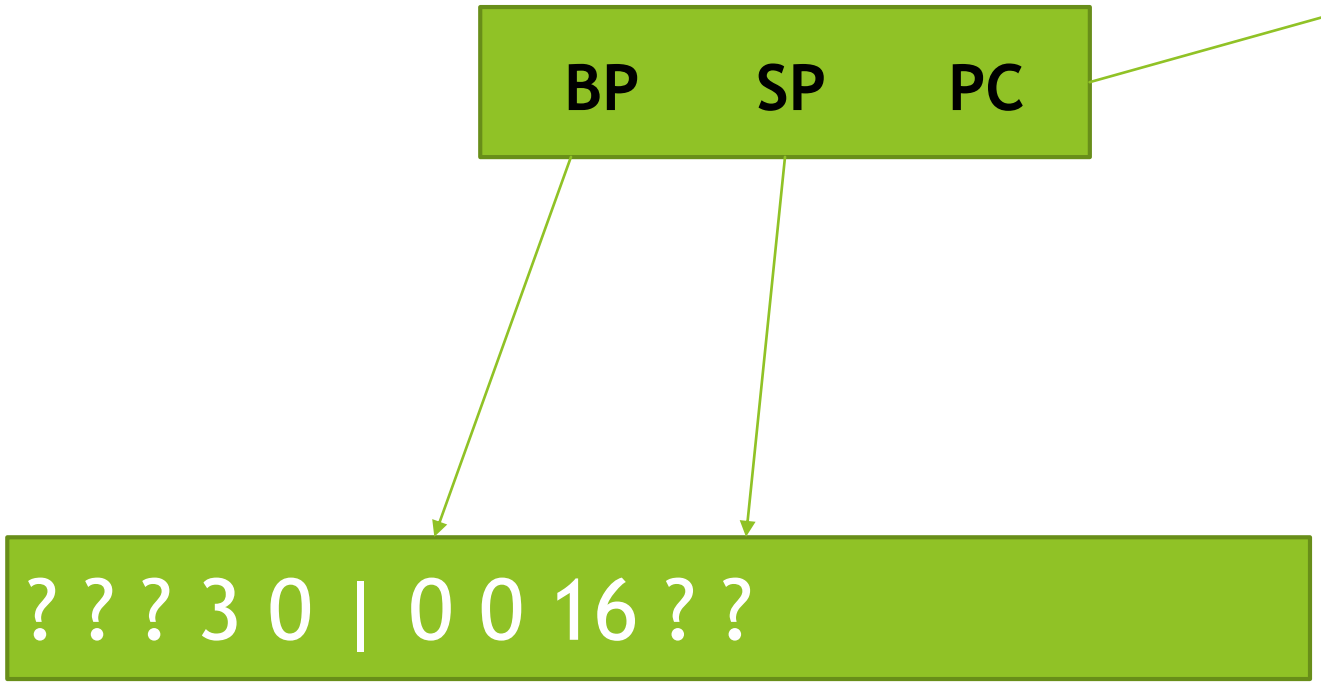
AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Note: In this example, both the static and dynamic link are the same. This is not always the case.

Execution

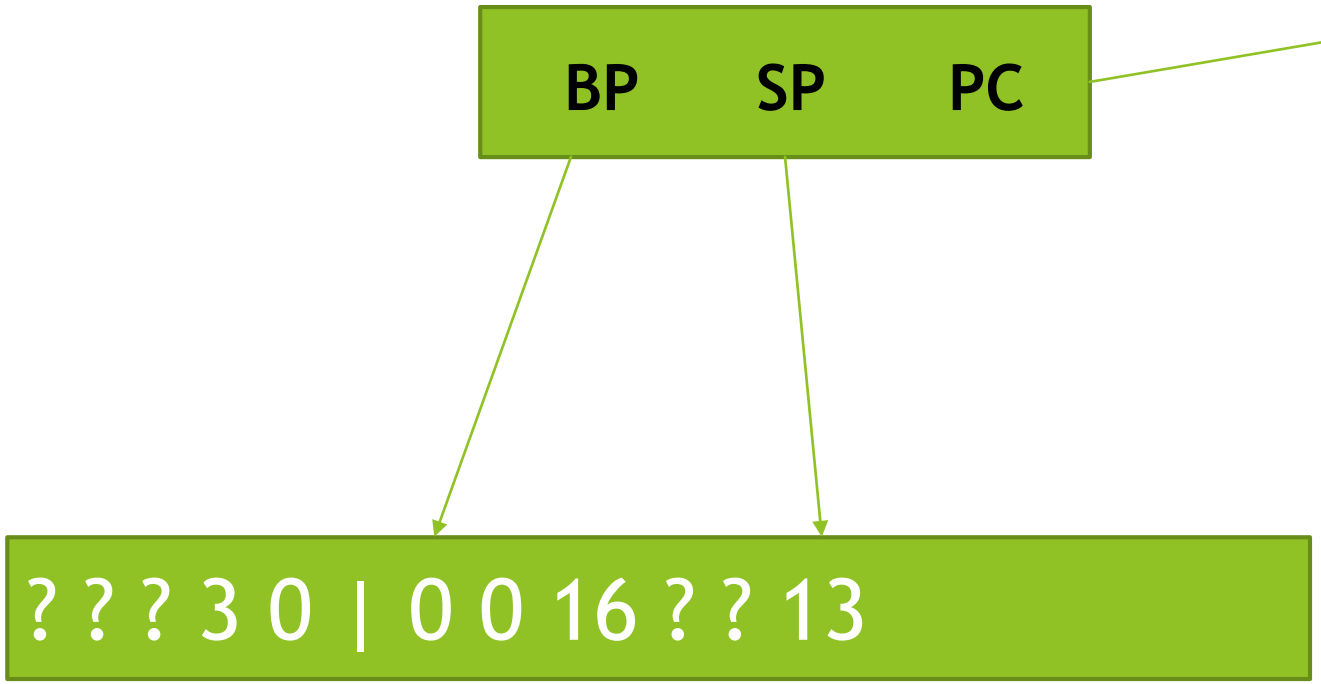


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution

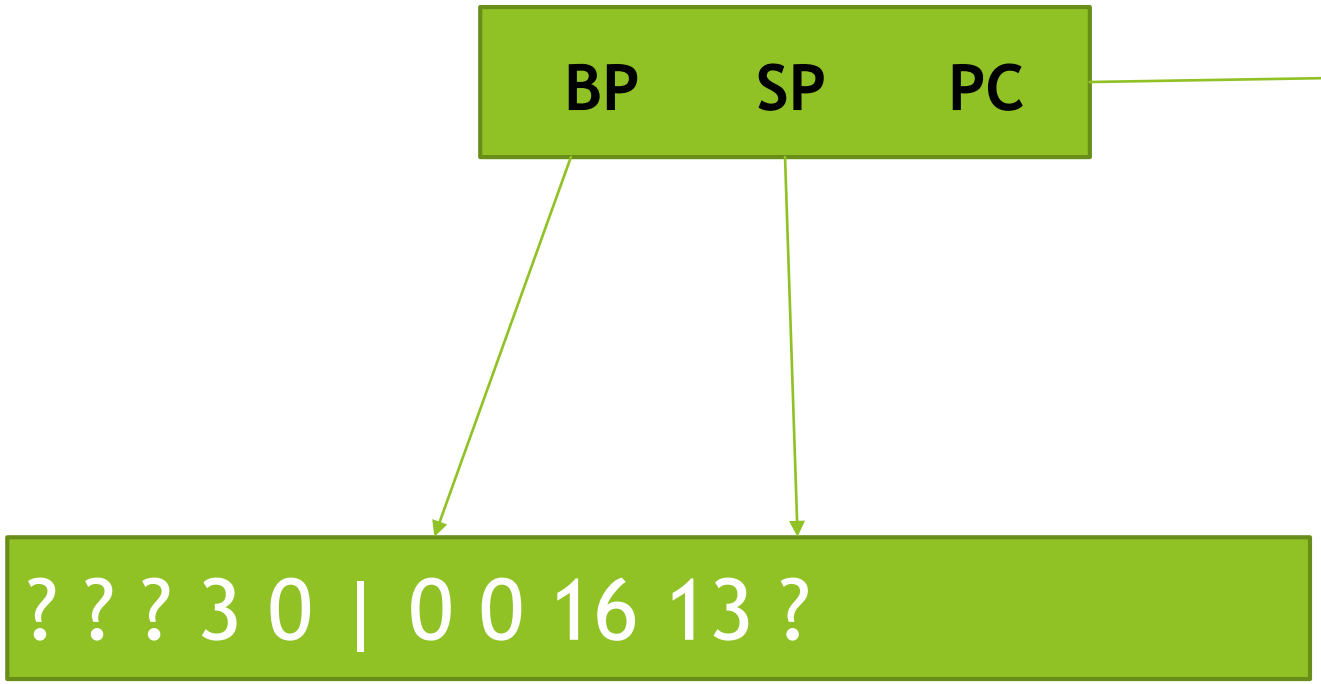


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution

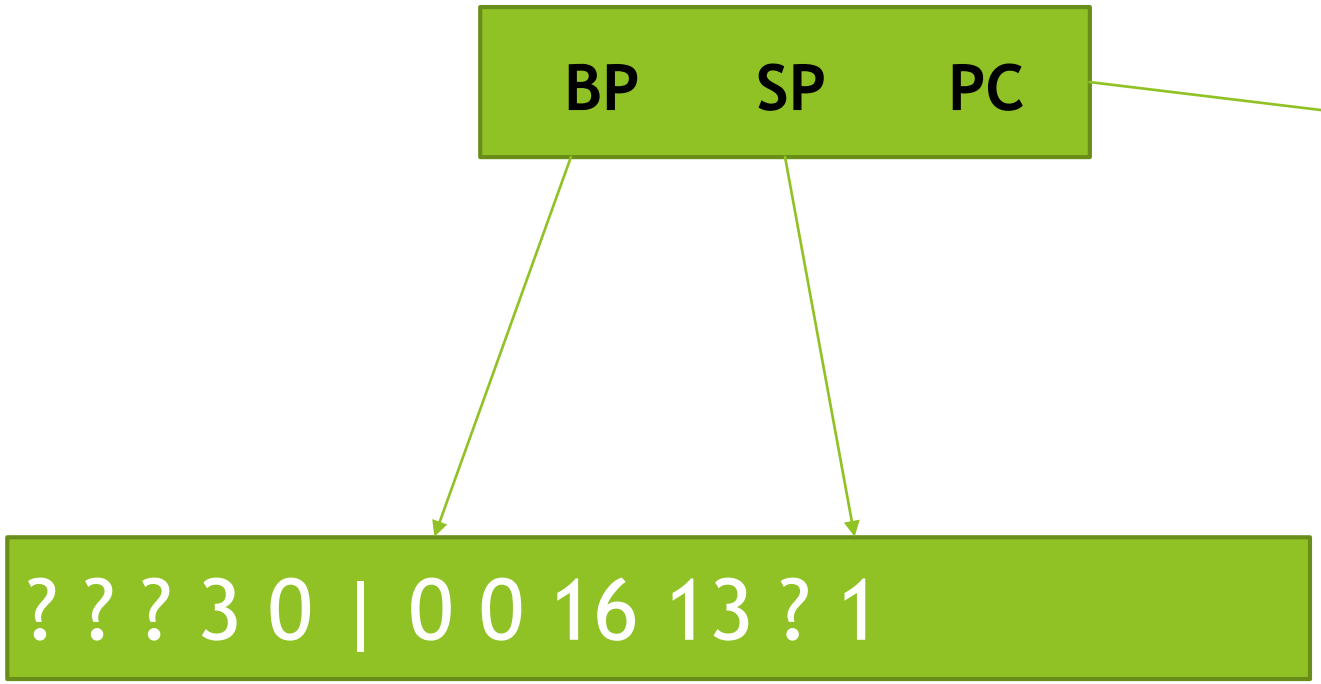


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution

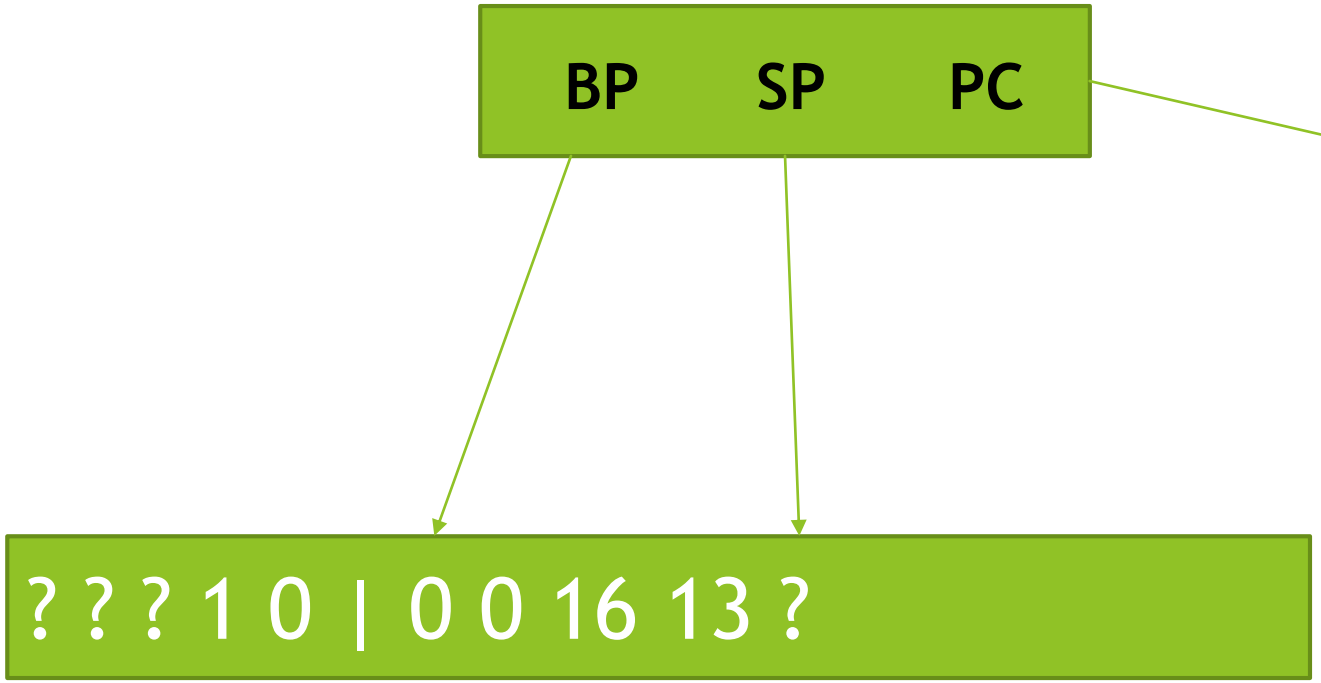


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution



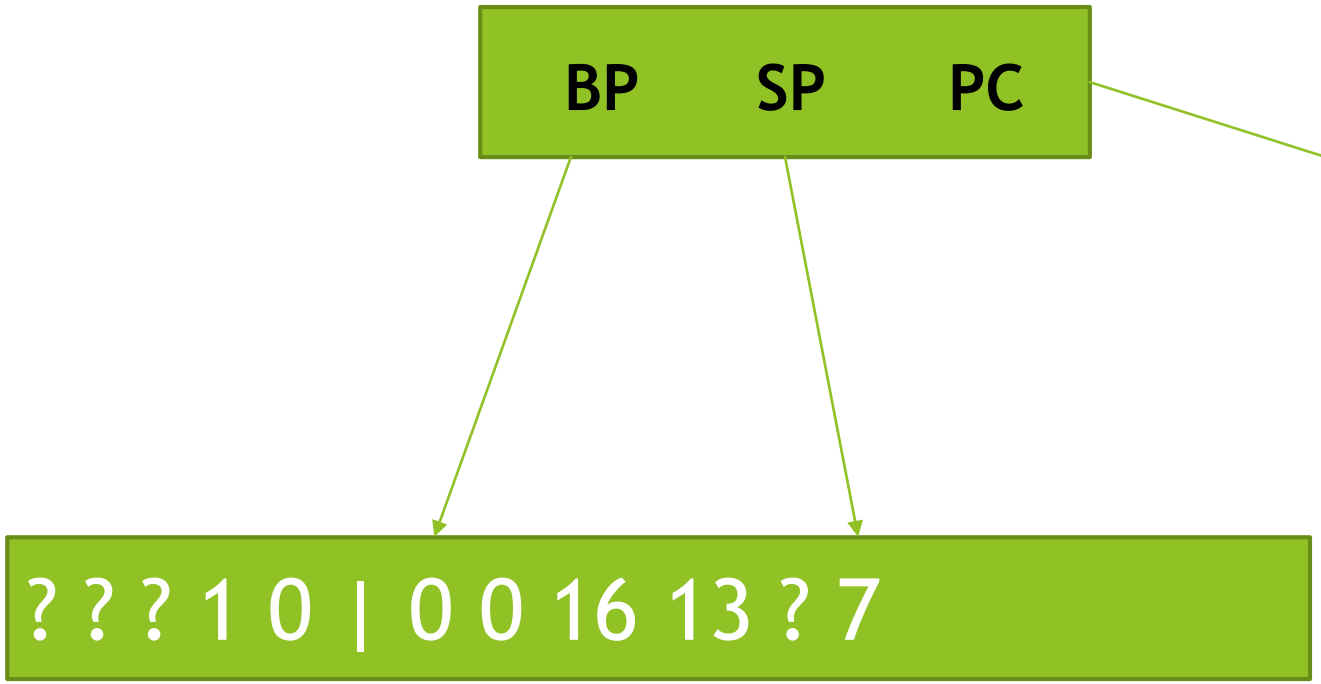
AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Note: base(L) means the base of the AR L static levels above the current AR

Execution

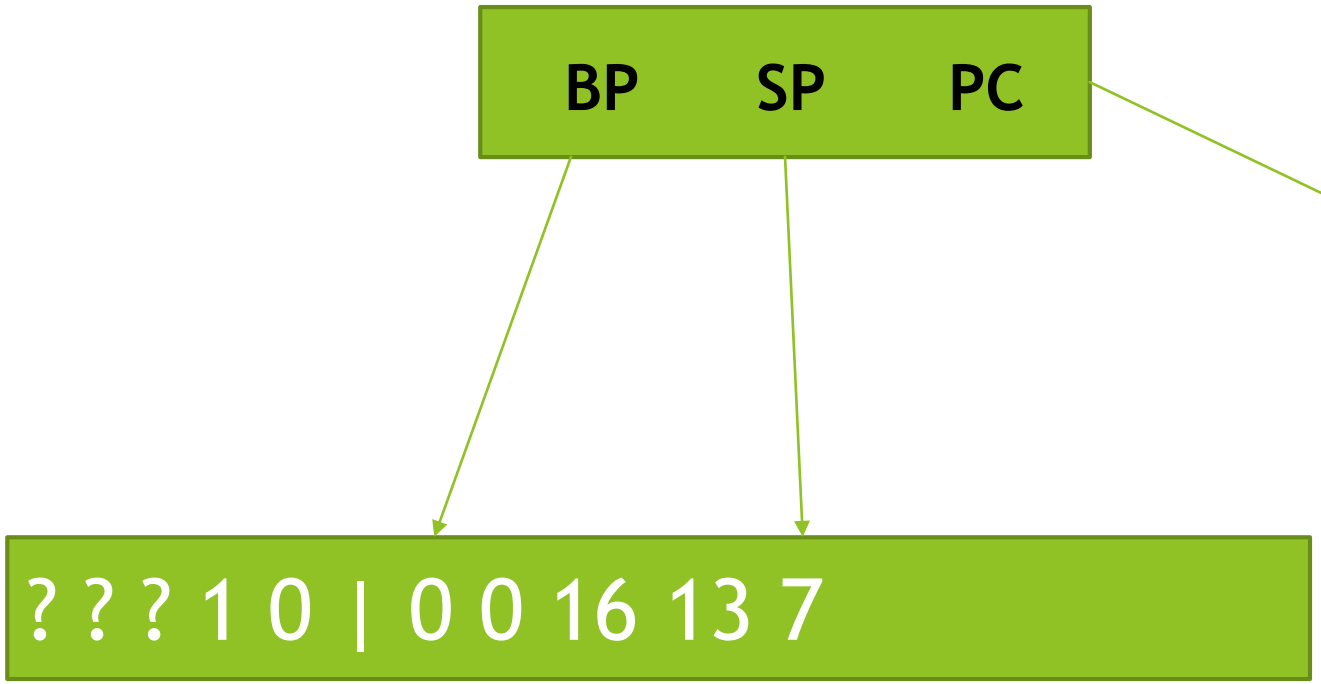


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution

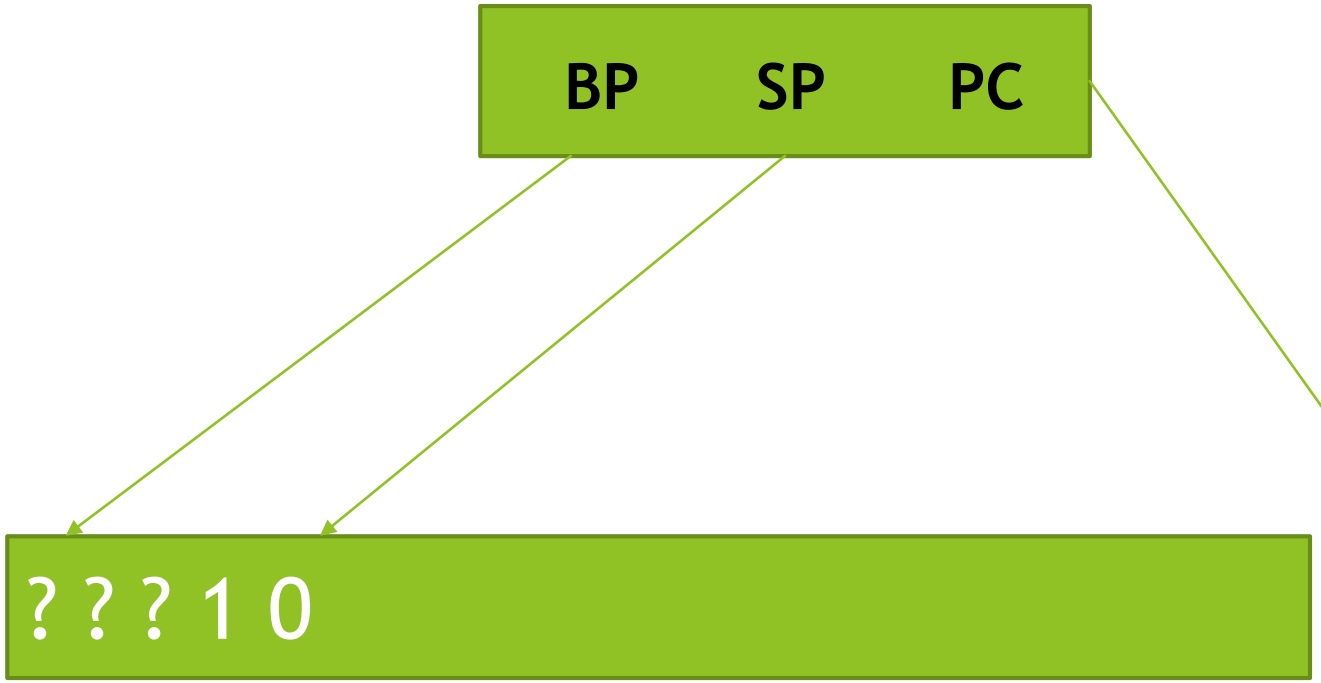


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution

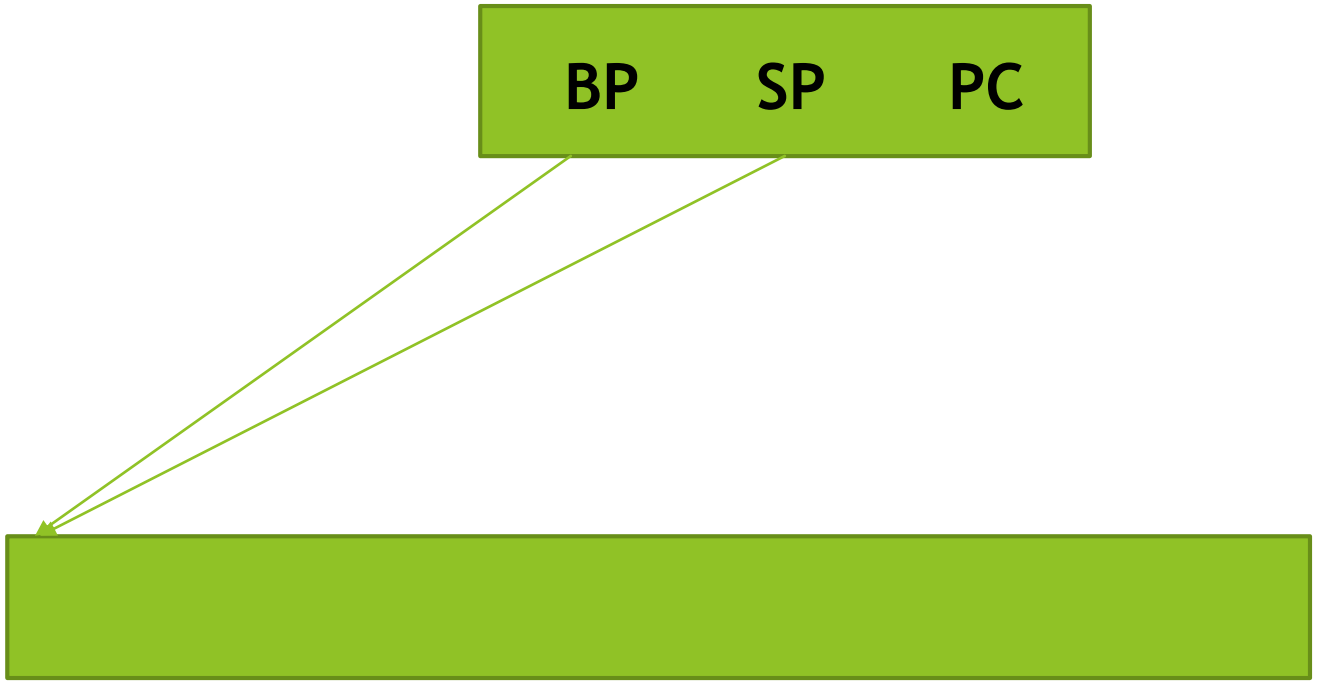


AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code

Execution



AR Stack

```
 0 JMP 0 10
 1 JMP 0 2
 2 INC 0 5
 3 LIT 0 13
 4 STO 0 3
 5 LIT 0 1
 6 STO 1 3
 7 LIT 0 7
 8 STO 0 4
 9 OPR 0 0
10 INC 0 5
11 LIT 0 3
12 STO 0 3
13 LIT 0 0
14 STO 0 4
15 CAL 0 2
16 OPR 0 0
```

Code