Lab 07

R. Ferrero, A. C. Marceddu

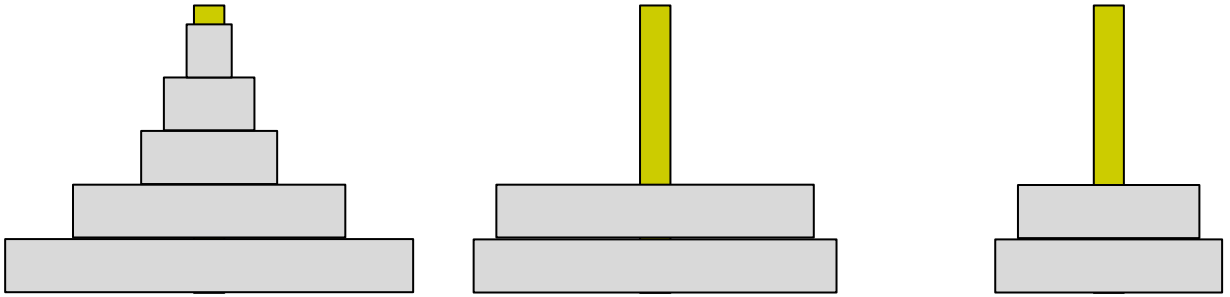
Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)

Torino - Italy

Tower of Hanoi

- The tower of Hanoi consists of 3 poles.
- Each pole contains a certain number of discs (possibly zero), arranged in descending order of size (the largest disc is the lowest).



Tower of Hanoi: implementation

- Each pole can be represented across a stack.
- The elements of the stack indicate the size of the discs contained in the corresponding pole.
- The 3 stacks are defined in a READWRITE area: the 3 stack pointers point to different memory locations within the data area.
- Freedom of choice is left to manage the 3 stacks as full descending or empty ascending.

Exercise 1

- Write a `fillStack` subroutine to initialize the stack associated to a pole.
- The subroutine receives two parameters:
  - the stack pointer associated to the pole
  - the address to a READONLY area containing a sequence of constants.
- The subroutine updates the two parameters:
  - the stack pointer points to the last entered data
  - the address in the READONLY area is subsequent to the last constant entered.

Exercise 1 (cont.)

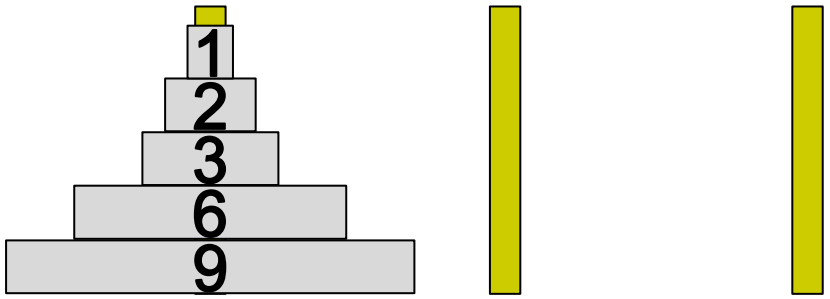
- The parameters are passed to the subroutine through the main stack (`SP`).
- The insertion ends when one of the following two conditions occurs:
  - the constant to be entered is 0
  - the constant to be inserted is greater than the last element on the stack.

Exercise 1: example

- `r1` is the stack pointer associated with the first pole
- The sequence of constants is

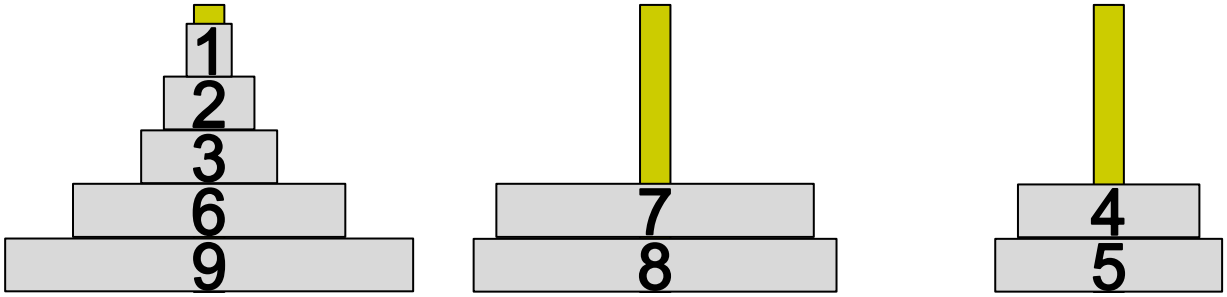
```
DCD 9, 6, 3, 2, 1, 8, 7, 0, 5, 4, 0
```

- `r0` contains the address of the first constant (9).
- After calling `fillStack` the situation is
  - $r1_{new} = r1_{old} \pm 20$
  - $r0_{new} = r0_{old} + 20$



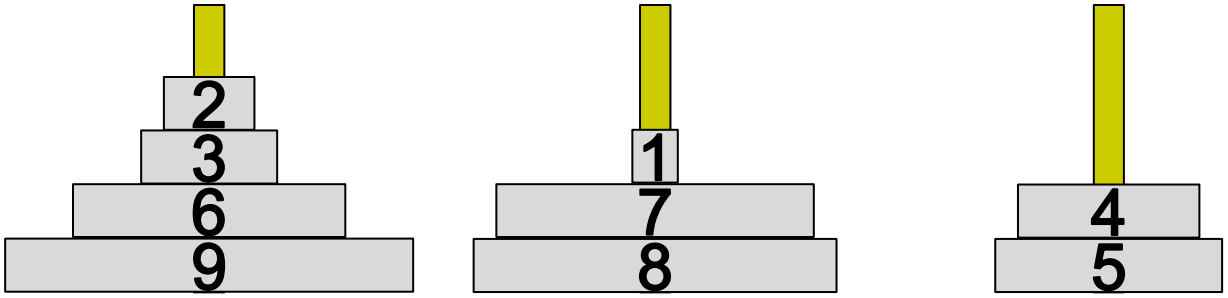
Exercise 1: example

- With three consecutive calls to `fillStack` (one for each pole) the tower of Hanoi becomes:



Tower of Hanoi: single movement

- It is possible to move a disc from one pole to another if one of two conditions occurs:
  - the new pole is empty
  - the upper disc in the new pole has a larger size than the disc to be moved.
- Example: a disc is moved from pole 1 to pole 2



Exercise 2

- Write a `move1` subroutine that handles moving a disk.
- `move1` receives 3 parameters across the stack
  - stack pointer of the starting pole
  - stack pointer of the destination pole
  - space for the return value
- `move1` returns 1 if it was possible to move the disk, 0 otherwise. Stack pointers are updated if the disk has been moved.

Exercise 2: warning

- To pass parameters to `move1`, t is preferable to use 3 PUSH instead of just one.
- Example: let `r1` be the stack pointer of the starting pole, `r2` the stack pointer of the destination pole, `r0` contain the return value.
- ```
  PUSH {r1, r2, r0}
  BL move1
  POP {r1, r2, r0}
  ```
  moves from pole `r2` to pole `r1`, because the registers are reordered as `{r2, r1, r0}`
  ```

Exercise 2: warning

- To ensure the movement from pole `r1` to pole `r2`, the following code can be used:

```
PUSH {r1}
PUSH {r2}
PUSH {r0}
BL move1
POP {r0}
POP {r2}
POP {r1}
```

Tower of Hanoi: multiple movements

- It is possible to move N discs from pole X to pole Y using a recursive procedure:
  - Move the first N-1 discs from X to Z
  - Move disk N from X to Y
  - Move N-1 discs from Z to Y

Exercise 3

- Write a `moveN` subroutine that manages the movement of N disks.
- `moveN` receives across the stack:
  - the stack pointer of the starting pole X
  - the stack pointer of the destination pole Y
  - the stack pointer of the auxiliary pole Z
  - the number of discs to move
- `moveN` updates the stack pointers and replaces the fourth parameter with the number of movements performed.

Pseudocode of moveN(X, Y, Z, N)

```
/* M is the number of mov. made*/
M = 0;
if (N == 1)
{
    move1(X, Y, a);
/*a is the return value: 0-1 */
    M = M + a;
}
```

Pseudocode of moveN(X, Y, Z, N)

```
else {
     moveN(X, Z, Y, N-1);
     M = M + b;
     move1(X, Y, a);
     if (a == 0) return;
     else M = M + 1;
     moveN(Z, Y, X, N-1);
     M = M + c;
}
```

Exercise 3: suggestions

- Since each recursive call adds its own parameters to the stack, it is suggested to increase the stack size by changing the value of `Stack_Size`.
- As a final check, considering a tower of Hanoi with N discs in one pole and 0 in the other two, note that the number of movements required to move the N discs into another pole is equal to $2^N - 1$.