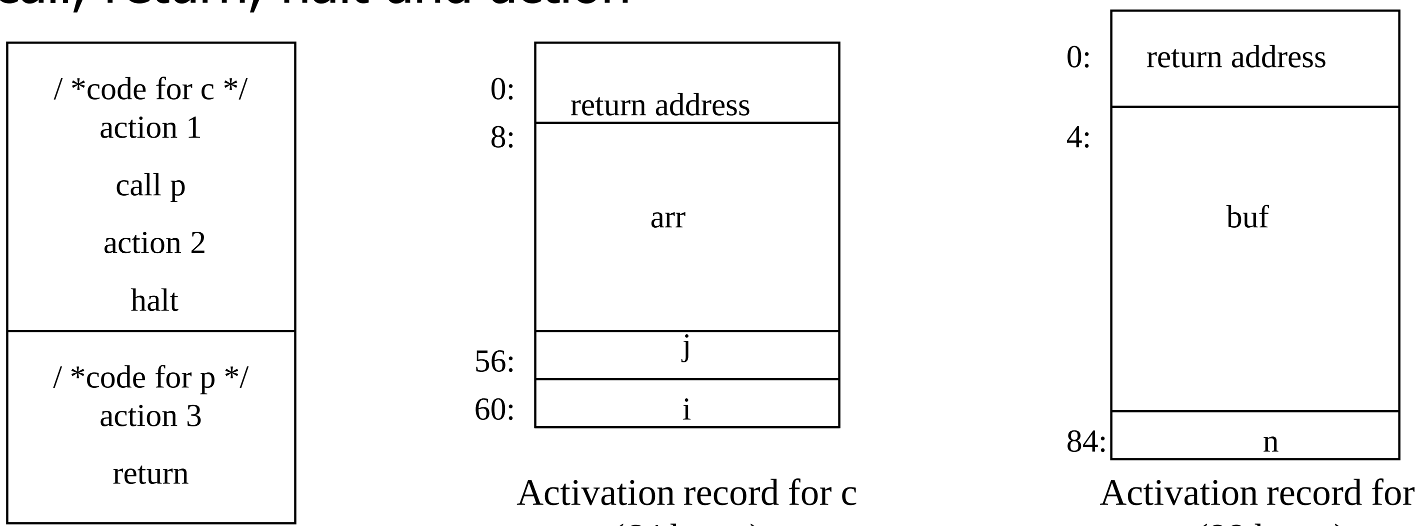
Run Time Storage Management

Run Time Storage Management

- Run time allocation and de-allocation of activations occurs as part of procedure call and return sequences

- Assume four kind of statements:
  call, return, halt and action



Three address code

Activation record for c (64 bytes)

Activation record for p (88 bytes)

Static Allocation

- A call statement is implemented by a sequence of two instructions

- A move instruction saves the return address

- A goto transfers control to the target code

- The instruction sequence is

  MOV #here+20, callee.static-area

  GOTO callee.code-area

Static Allocation Cont …

- callee.static-area and callee.code-area are constants referring to address of the activation record and the first address of called procedure respectively.

- #here+20 in the move instruction is the return address; the address of the instruction following the goto instruction

- A return from procedure callee is implemented by

  GOTO *callee.static-area

Example

- Assume each action block takes 20 bytes of space
- Start address of code for c and p is 100 and 200
- The activation records are statically allocated starting at addresses 300 and 364.

100: ACTION I
120: MOV #140, 364
132: GOTO 200
140: ACTION 2
160: HALT
:
200: ACTION-3
220: GOTO *364
:
300:
304:
:
364:
368:

Stack Allocation

- Position of the activation record is not known until run time

- Position is stored in a register at run time, and words in the record are accessed with an offset from the register

- The code for the first procedure initializes the stack by setting up SP to the start of the stack area

  MOV #Stackstart, SP
  code for the first procedure
  HALT

Stack Allocation Cont…

- A procedure call sequence increments SP, saves the return address and transfers control to the called procedure

```
ADD #caller.recordsize, SP
MOVE #here+ 16, *SP
GOTO callee.code_area
```

Stack Allocation Cont…

- The return sequence consists of two parts.
- The called procedure transfers control to the return address using

  GOTO *0(SP)

  0(SP) is the address of the first word in the activation record and *0(SP) is the return address saved there.

- The second part of the return sequence is in caller which decrements SP

  SUB #caller.recordsize, SP

Example

- Assume activation records for procedures s, p and q are ssize, psize and qsize respectively (determined at compile time)
- First word in each activation holds the return address
- Code for the procedures start at 100, 200 and 300 respectively, and stack starts at 600.

```
action-l    /* code for s * /
call q
action-2
halt

action-3    /* code for p * /
return

action-4    /* code for q * /
call p
action-5
call q
action-6
call q
return
```

Example

```
100: MOVE #600, SP
108: action-1
128: ADD #ssize, SP
136: MOVE 152, *SP
144: GOTO 300
152: SUB #ssize, SP
160: action-2
180: HALT
. . .

200: action-3
220: GOTO *0(SP)
. . .
```

```
300: action-4
320: ADD #qsize, SP
328: MOVE 344, *SP
336: GOTO 200
        344: SUB #qsize, SP
352: action-5
372 ADD #qsize, SP
380: MOVE 396, *SP
388: GOTO 300
396 SUB #qsize, SP
404: action-6
424: ADD #qsize, SP
432: MOVE 448, *SP
440: GOTO 300
448: SUB #qsize, SP
456: GOTO *0(SP)
```