Outline of Lecture

Supporting Procedures in MIPS

Supporting Procedures in Computer Hardware

- ***Subroutines*** are necessary in any programming language for better structuring of programs - thus we must have an instruction that jumps to a procedure and then returns from the procedure.

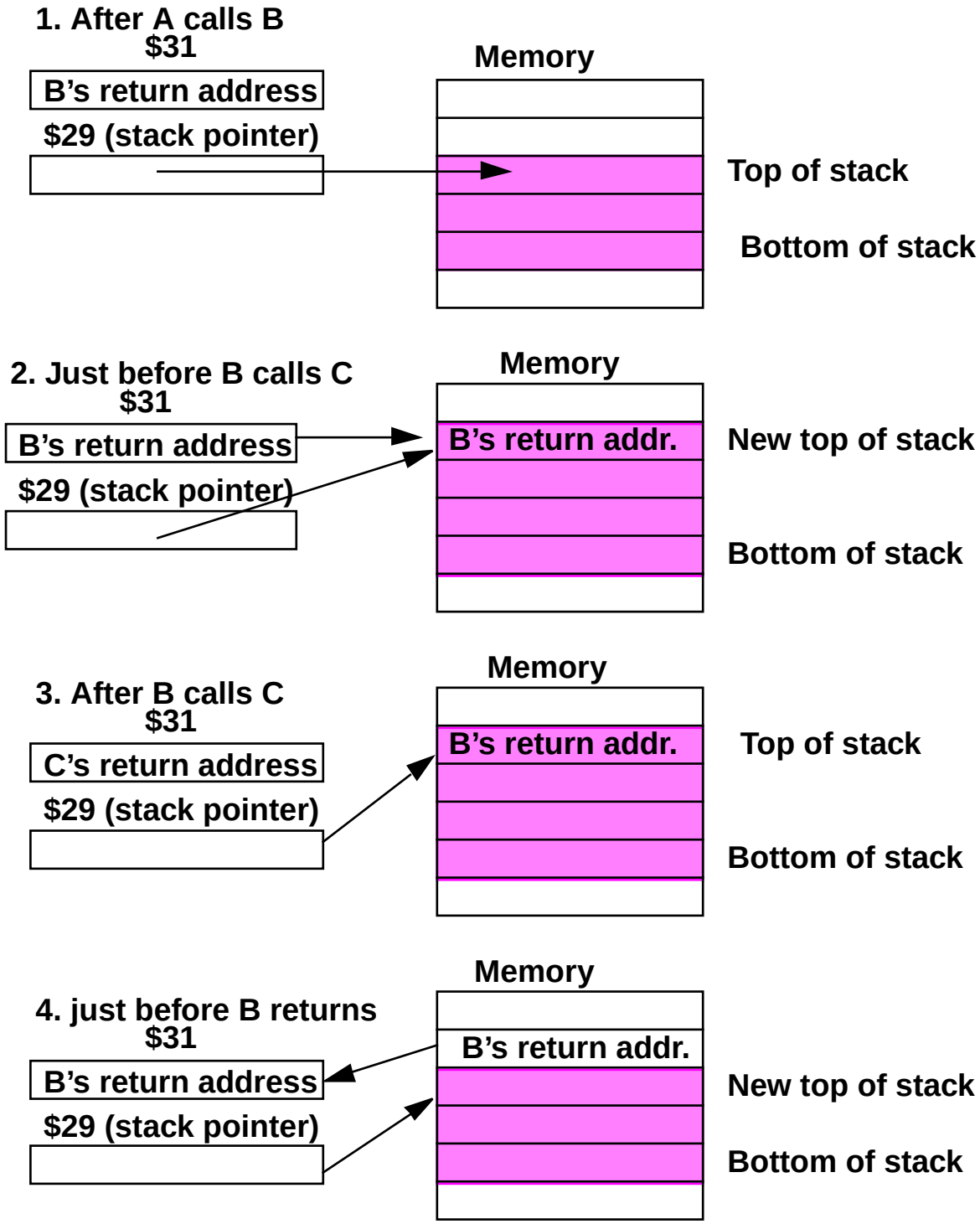
- The MIPS assembly language provides an instruction that jumps to an address and simultaneously saves the address of the following instruction in register $31 - it is the `jump-and-link (jal)` instruction (e.g., `jal ProcedureAddress`).

- The address stored in register $31 is called the `return address`. The parameters of the procedure are passed in registers $4 through $7.

- Register $31, which points to the instruction being executed in the program, is called **program counter** (**PC**).

- Since procedures can call other procedures, we need to save the values stored in the registers before jumping to a procedure. The ideal way to save these registers is in a **stack (last-in-first-out).** Instructions for operating on the stack are called **push** and **pop**.

- The detailed steps for manipulating the stack and the PC when calling procedures are as follows:


1. After A calls B
$31
B's return address
$29 (stack pointer)
Memory
Top of stack
Bottom of stack
2. Just before B calls C
$31
B's return address
$29 (stack pointer)
Memory
B's return addr.
New top of stack
Bottom of stack
3. After B calls C
$31
C's return address
$29 (stack pointer)
Memory
B's return addr.
Top of stack
Bottom of stack
4. just before B returns
$31
B's return address
$29 (stack pointer)
Memory
B's return addr.
New top of stack
Bottom of stack

MIPS Addressing Modes

- The MIPS architecture provides two more ways of accessing operands:

Constant / Immediate Operands

- ***Constants*** are used frequently in all kinds of programs (50% - 70% of arithmetic operands are constants). For example to add 4 to register $sp:

```
lw $t0, AddrConstant4($zero) # $t0=constant 4

add $sp, $sp, $t0 # $sp = $sp + $t0
```

- A better way is to avoid memory access, and offer new versions in which the operand can be a constant - this is the **I-type** (**`immediate`**) instruction format.

Example

The add instruction that has one constant operand is called `add immediate` or `addi`. To add 4 to register $29 we just write

```
addi $sp, $sp, 4 # $sp = $sp + 4
```

What is the corresponding MIPS machine code?

Answer

The MIPS machine code is as follows:

op	rs	rt	immediate
8	29	29	4

op	rs	rt	immediate
001000	11101	11101	0000 0000 0000 0100

- We can also compare the values to a constant using the instruction **set less than immediate** (**slti**):

```
slti $t0, $s2, 10  #  $t0 = 1 if $s2 < 10
```

Principle #4: ***Always make the common case faster*****.**

Since constants are frequently used, then it is very beneficial to have an immediate addressing mode.

Further Reading

Chapter 3 and Appendix. **David A. Patterson and John L. Hennessy. *Computer Organization & Design: The Hardware / Software Interface*. Morgan Kaufman Publishers, 1998.**