High Performance Computing
Lecture 8

Matthew Jacob

Indian Institute of Science

Example: Function Call and Return

What must be done on a function call?

- Pass parameters on stack
- Transfer control to start of function
- Remember return address
  - Where? On a stack (in main memory)
- Save register values on stack
- Allocate space for local variables on stack

What must be done on a function return?

- Pass return value (through stack)
- Restore register values from the stack
- Clean up stack
- Transfer control back to return address

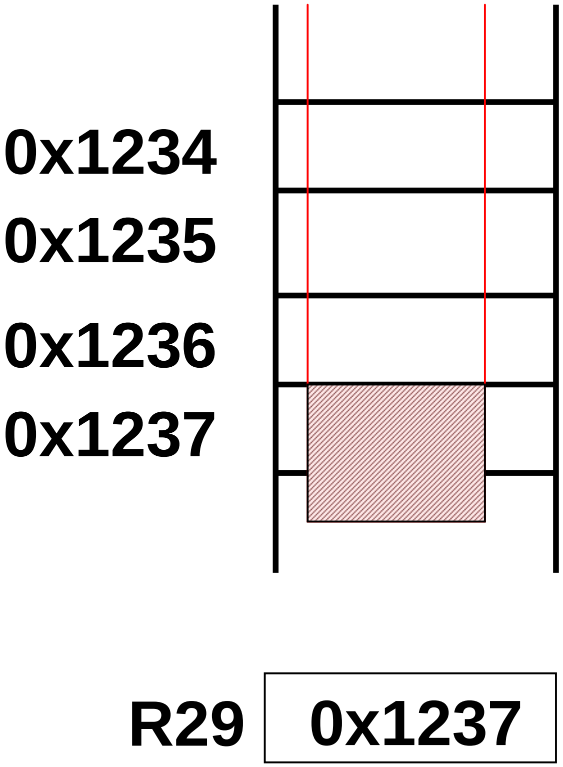
Implementing a Stack in Memory

- Use one register as Stack Pointer, say R29
  - It could point at either
    - The current top of stack value, or
    - The memory location for the next push onto the stack
- Decide whether stack grows “up” or “down” in memory
  - up: grows into higher memory addresses
  - down: grows into lower memory addresses

Implementing a Stack in Memory.

Example: Growing down (into lower addresses) in memory

R29 pointing at current top of stack element



0x1234

0x1235

0x1236

0x1237

R29 0x1237

```
PushByte:   SUBI  R29, R29, 1
            SB     0(R29), Rs

PopByte:    LB    Rd, 0(R29)
            ADDI R29, R29, 1
```

Function Call and Return

```
void B (int x) {
   int a, b;
     …

     return();
}
```

```
void A() {
     …
     B(5);
     …
}
```

B:

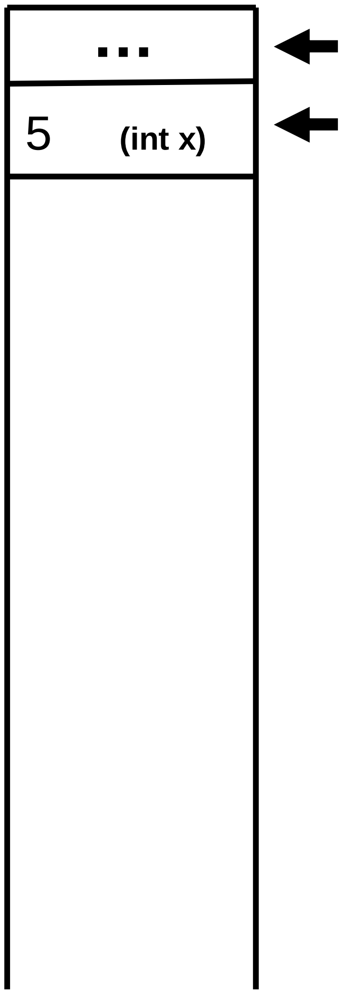
ADDI R1, R0, 5

ADDI R29, R29, 4

SW 0(R29), R1

JAL B

Function Call/Return Stack



Recall: MIPS 1 JAL instruction

	Mnemonics	Example	Meaning
Conditional Branch	BEQ, BNE, BGEZ, BLEZ, BLTZ, BGTZ	BLTZ R2, -16	If R2 < 0, PC ← PC + 4 -16
Jump	J, JR	J target_{26}	PC ← $(PC)_{31-28} \| \| \text{target}_{26} \| \| 00$

Jump and Link	JAL, JALR	JALR R2	R31 ← PC + 8
			PC ← R2

System call	SYSCALL	SYSCALL	

Function Call and Return

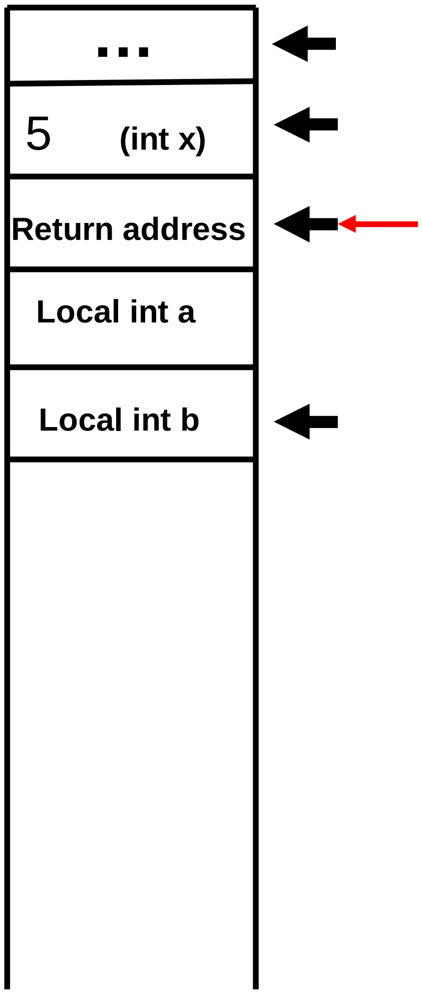
```
void B (int x) {
   int a, b;
     …


     return();
}
```

```
void A() {
     …
     B(5);
     …
}
```

```
B:      ADDI R29, R29, 4
        SW 0(R29), R31
        ADDI R29, R29, 8
        …
        SUBI R29, R29, 16
        LW R31, 8(R29)
        JR R31
```

```
ADDI R1, R0, 5
ADDI R29, R29, 4
SW 0(R29), R1
JAL B
```

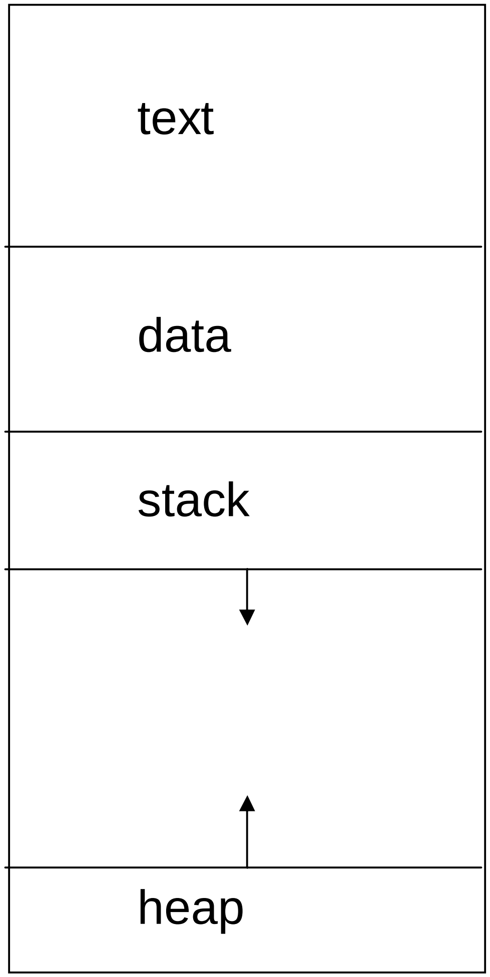
Function Call/Return Stack



…
5 (int x)
Return address
Local int a
Local int b

Use of Main Memory by a Program

- Instructions (code, text)
- Data used in different ways
  - Stack allocated
  - Heap allocated
  - Statically allocated



Use of memory addresses

Stack Allocated Variables

- Space allocated on function call, reclaimed on return
- Addresses calculated and used by compiler, relative to the top of stack, or some other base register associated with the stack
- Growth of stack area is thus managed by the program, as generated by the compiler

Heap Allocated Variables

- Managed by a memory allocation library
- Functions like malloc, realloc ,free
- Get `linked’ (joined) to your program if they are called
- Executed just like other program functions
- What about growth of the heap area?
  - Managed by the library functions