

Procedures

Procedures are very important for writing reusable and maintainable code in assembly and high-level languages. How are they implemented?

- Application Binary Interfaces
- Calling Conventions
- Recursive Calls
- Examples

Reference: PowerPC Embedded ABI

General Concepts

- Caller: The calling procedure

Callee: The procedure called by the caller

...	int mult(x, y)
prod = mult (a, b)	...
...	return (x * y)

- Caller and callee must agree on:
 - How to pass parameters
 - How to return the return value(s), if any
 - How to maintain relevant information across calls
- PowerPC architecture does not define “agreement”. Instead, common policies are defined by convention.

PowerPC Features

The PowerPC ISA provides the following features to support procedure/function calls:

- link register (p. 2-11)
- bl: branch and link (p. 4-41)
- blr: branch to link register (Table F-4)

A Very Simple Calling Convention

- Passing arguments
 - Use GPRs r3 to r10 in order
 - Use stack in main memory if more than 8 arguments
- Passing return value
 - Leave result in r3

Example

```
int func(int a, int b)
{
    return (a + b);
}
```

```
main
{
    ...
    func(5,6);
    ...
}
```

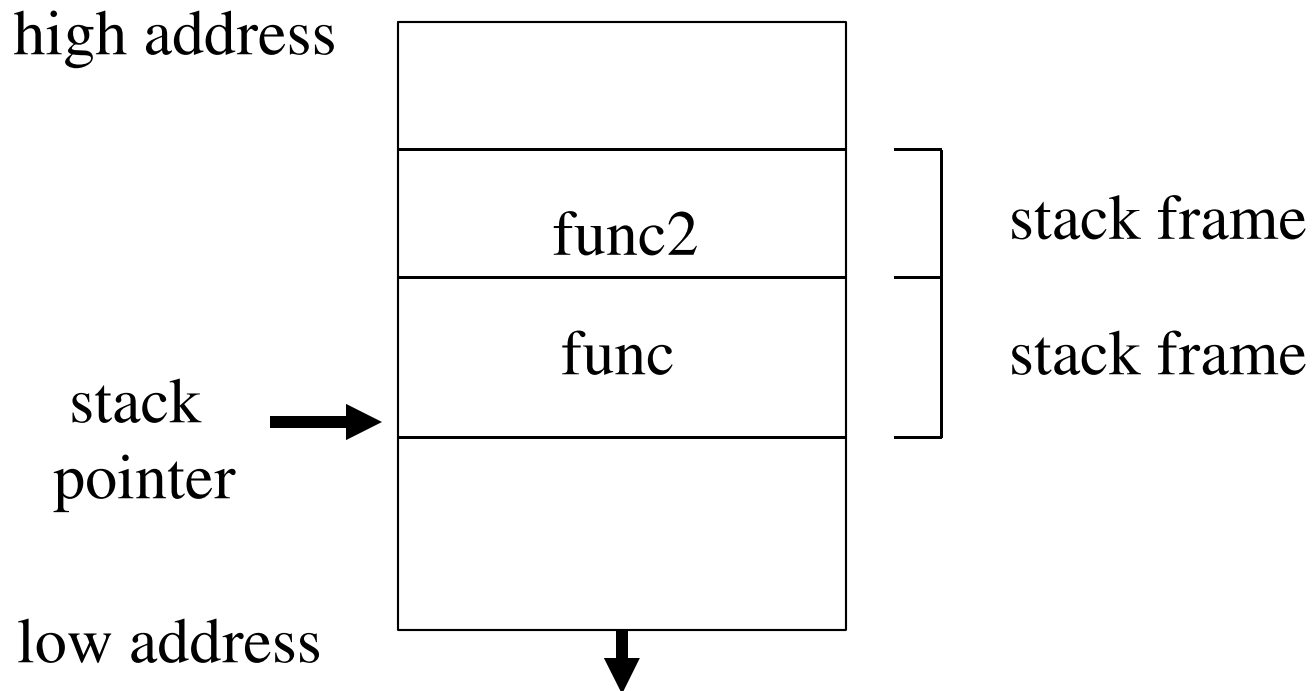
Another Example

```
int func2(int a, int b)
{
    return func(a , b);
}
```

```
main
{
    ...
    func2(5,6);
    ...
}
```

The Stack

- Information for each function invocation (e.g. link register) is saved on the *call stack* or simply *stack*.
- Each function invocation has its own *stack frame* (a.k.a. *activation record*).



Using the Stack

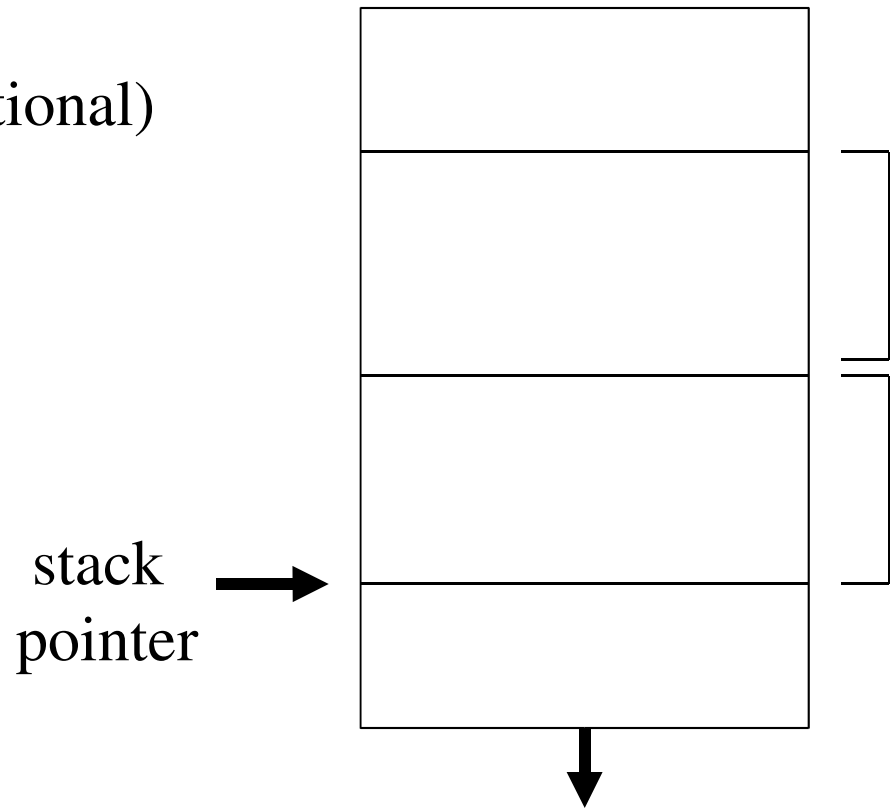
main ...
...
bl func2
...
func2 ...
...
bl func
...
??
...
func ...
...
??

Describe the stack and LR contents

- right before the first bl
- right after the first bl
- right after the second bl
- after each procedure returns

What Goes into Stack Frame?

- link register
- passing parameters (optional)
- return value (optional)
- what else?



Application Binary Interface (ABI)

- Application Binary Interface (ABI): Defines everything needed for a binary object file to run on a system (CPU and operating system), so that it can
 - call system library routines
 - call (and be called by) code generated by other people and other compilers
- The ABI specifies:
 - file format
 - rules for linker operation
 - procedure calling conventions
 - register usage conventions
- PowerPC has different but very similar ABIs for MacOS, AIX, embedded systems (EABI), Unix, Windows.

PowerPC Conventions

- Stack pointer is r1
- Stack pointer is double-word aligned (8 bytes)
- Stack “grows” down, toward address 0
- r1 points to lowest stack address (bottom of current stack frame)
- First item in stack frame (offset 0 from r1) is address of previous stack frame (a.k.a. *frame pointer*)
- Second item (offset 4) is saved link register
- Minimum stack frame is 8 bytes
- Stack frame optional for leaf functions
- Always use update addressing mode to allocate stack frame automatically

Register Usage Convention

Rules about who gets to use/save which registers

- Caller-save: r0, r3-r12, LR, CTR, CR0, CR1, CR5-7
- Callee-save: r14-r31, CR2-4 (non-volatile registers whose values must be preserved across calls and must thus be restored by callee before returning to caller)

Dedicated: r1, r2, r13

EABI Stack Frame

