The Stack

Embedded Software Essentials

C1M3V6

Data Segment

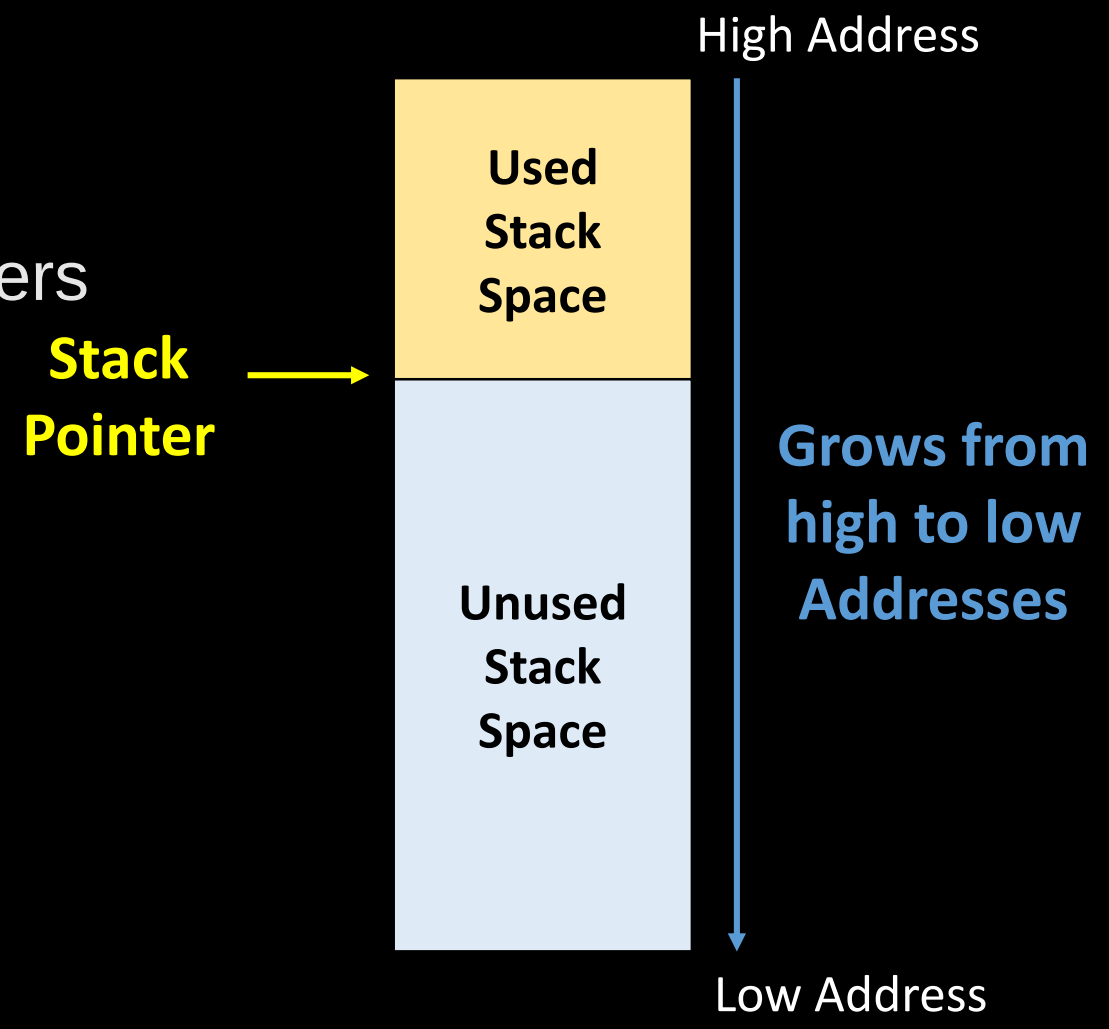
- Four Main Segments
  - Stack
  - Heap
  - Data
  - BSS

```
int foo(int D_STACK_REG){
  int a_STACK;
  char * ptr_STACK;

  /* More Code Here */

  return a_STACK;
}
```

- Stack space is reserved at compile time, data is allocated at runtime by precompiled instructions

The Stack

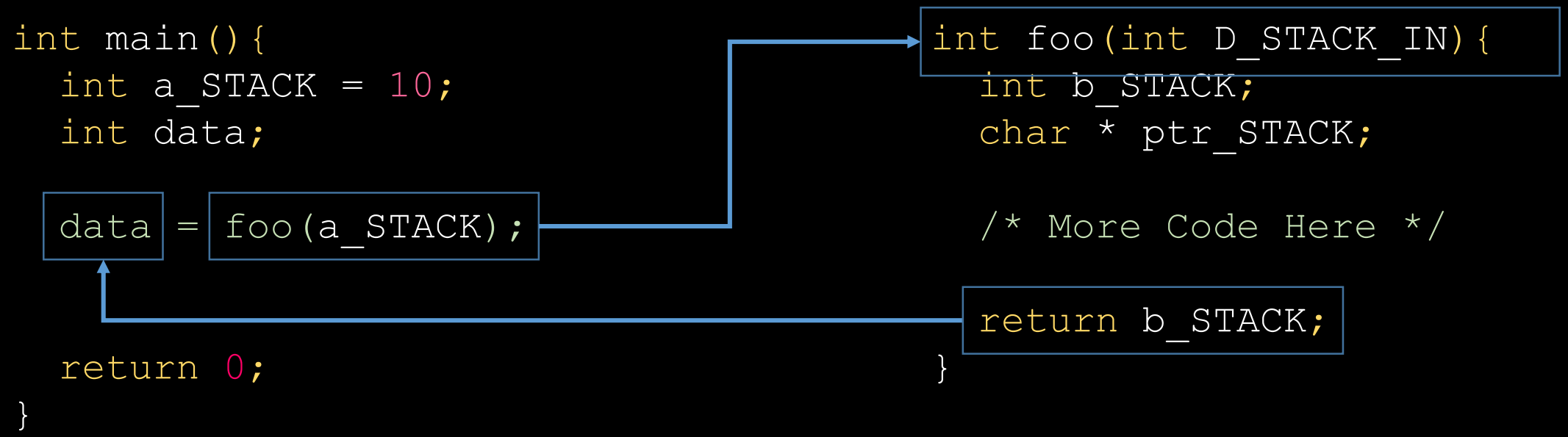
- General Stack Characteristics
  - Stack Specific Instructions and Registers
  - Ascending or Descending Growth
  - Architecture Specific Contents
  - Content Order
- Stack segment is reserved at linking step
  - Compile Option/Linker File specifies size
- Stack memory is reused throughout program
  - Stores “temporary” data



Calling Convention

- **Calling Convention**: Describes the method of how data is passed in and returned from a routine

Execution Flow changes: copy params, save state, jump into sub-routine

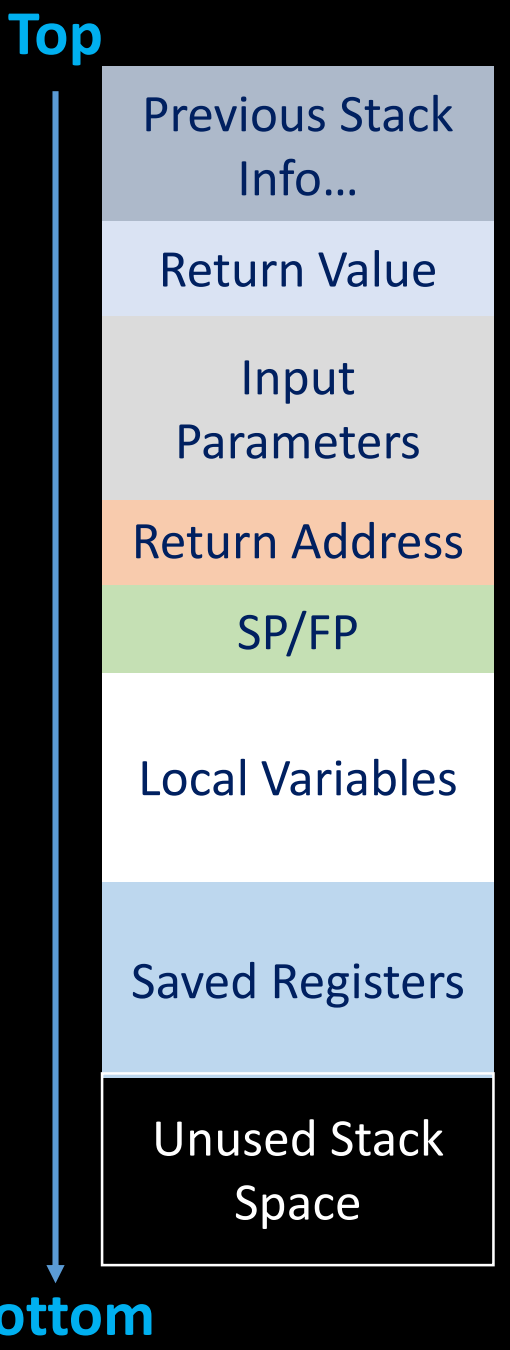


Remove sub-routine allocations, return data, restore calling function state

Stack Implementations

- Typically stores (Depends on Architecture):
  - Local Variables
  - Input Parameters
  - Return Data
  - Copy of Used Registers
  - Return Address
  - Previous Stack Pointer
  - Copy of Special Function Registers (Interrupts)
- Has a set size (compiler option or linker file)

Grows and shrinks at one end[1]

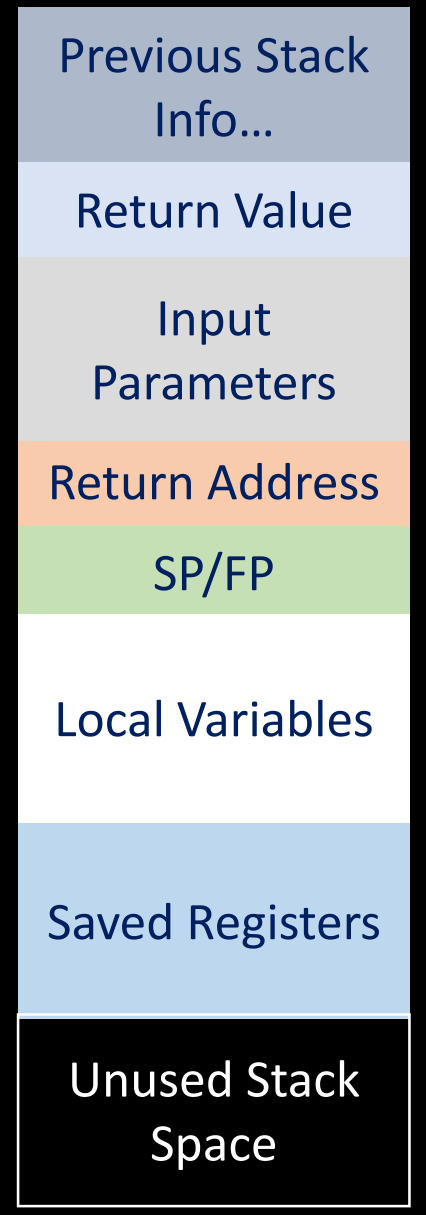


[1]Example gives an arbitrary order of info

Stack Frame

- Local Variables
- Input Parameters
- Return Data
- Used Registers
- Return Address
- Previous Stack Pointer
- Special Function Registers

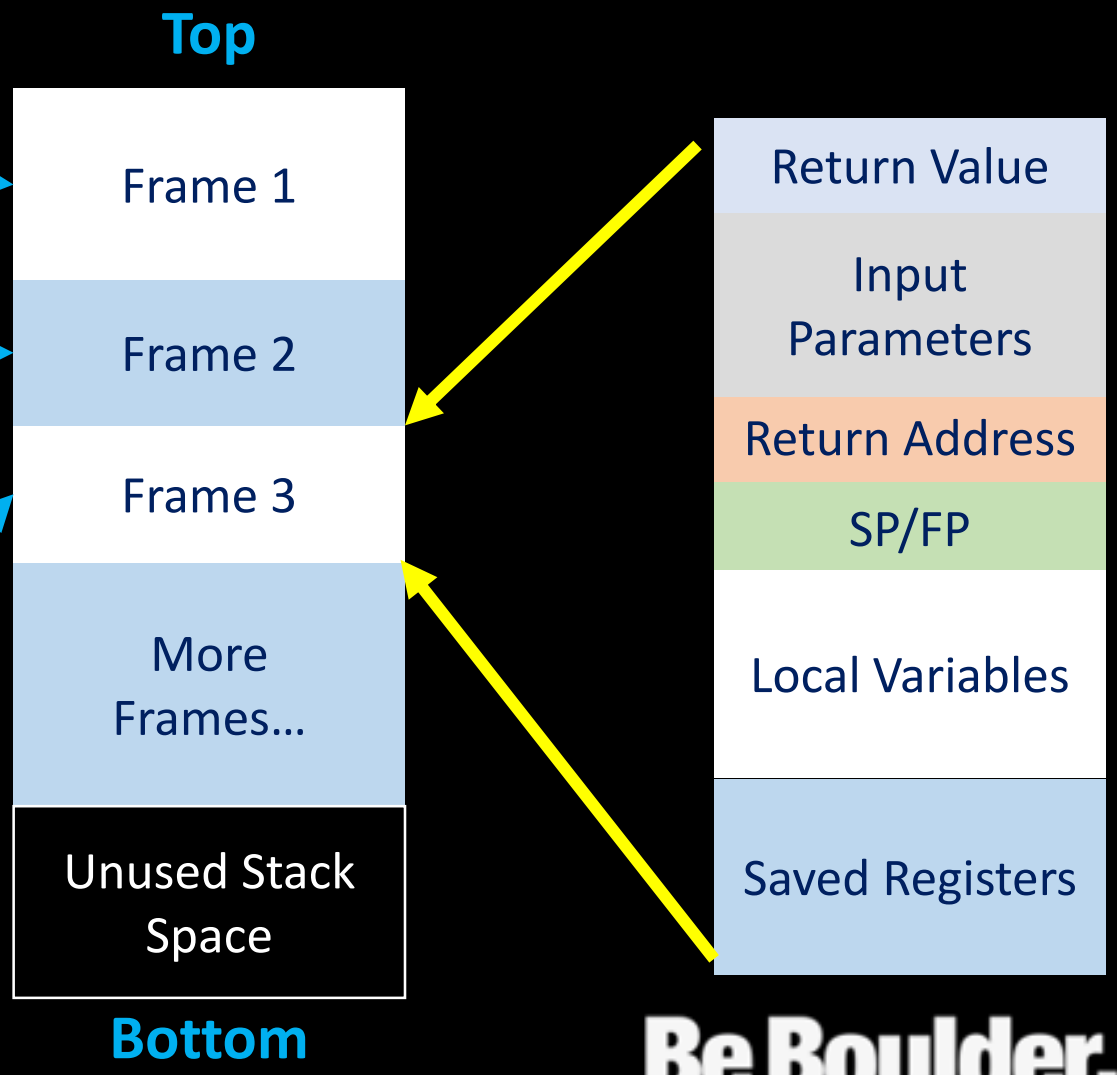
Stack Frame



Frame Size Depends on Routine Implementation!!!

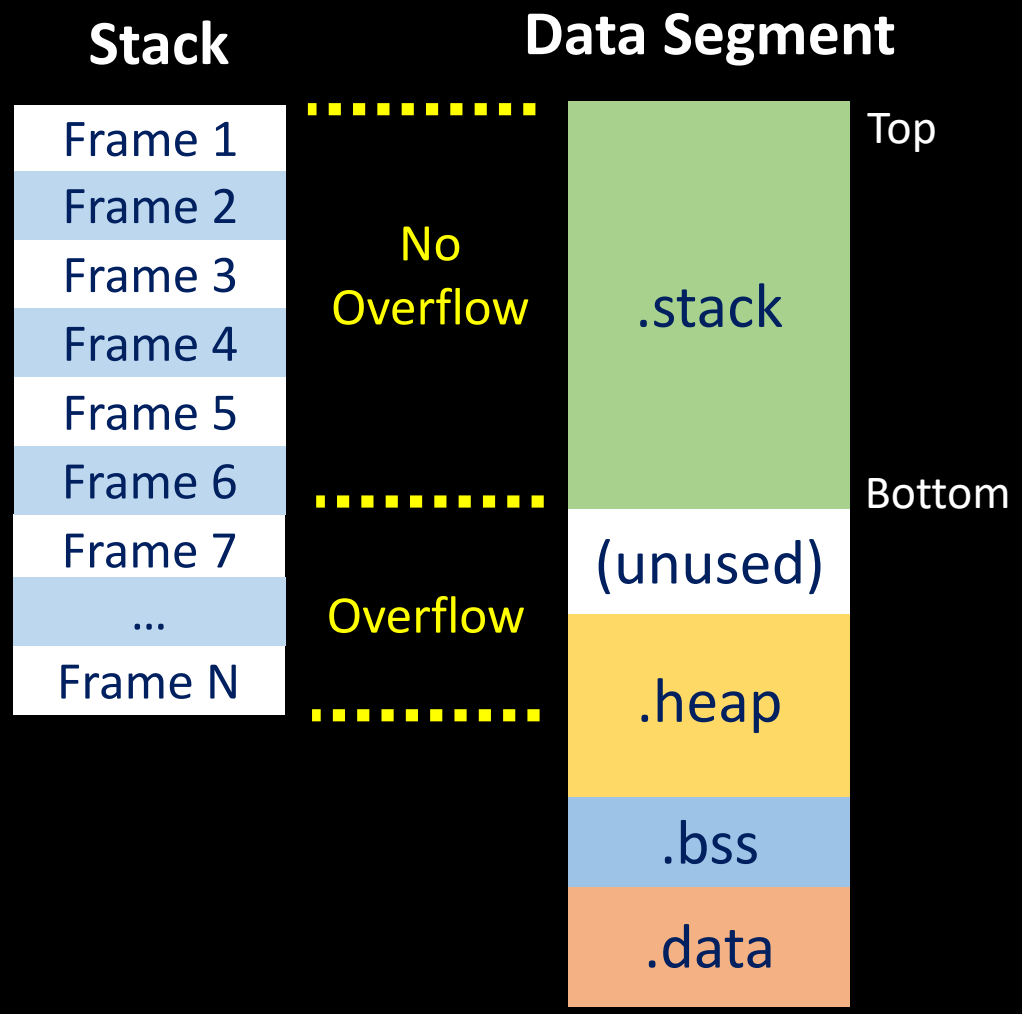
Stack Frames

```
int main(){
  int a_STACK = 10;
  int data;

  data = foo(a_STACK);

  return 0;
}

int foo(int D_STACK_IN){
  int b_STACK;
  char * ptr_STACK;

  bar(b_STACK, ptr_STACK);

  return b_STACK;
}
```



Stack Frames

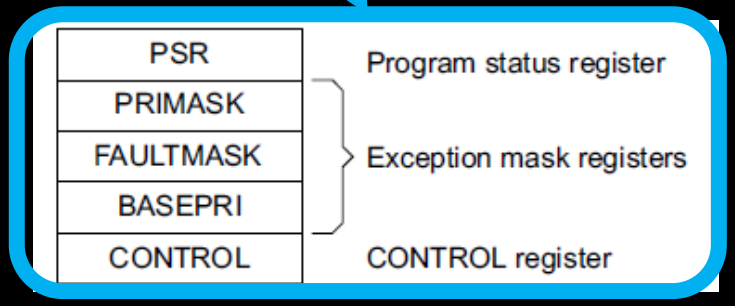
- Frame size is affected by:
  - Number of Local Variables
  - Number of Input Parameters
  - Size/Type of Local Variables
  - Size/Type of Input Parameters
  - Size/Type of Return Data
  - Number of Nested Subroutine calls
  - Interrupts/Nested Interrupts



Depends on Architecture/Application Binary Interface's Calling Convention

ARM Registers and the Stack

Special Purpose Registers



General Purpose Registers

Register	Synonym	Special	Role in the procedure call standard
r15		PC	The Program Counter.
r14		LR	The Link Register.
r13		SP	The Stack Pointer.
r12		IP	The Intra-Procedure-call scratch register.
r11	v8		Variable-register 8.
r10	v7		Variable-register 7.
r9		v6 SB TR	Platform register. The meaning of this register is defined by the platform standard.
r8	v5		Variable-register 5.
r7	v4		Variable register 4.
r6	v3		Variable register 3.
r5	v2		Variable register 2.
r4	v1		Variable register 1.
r3	a4		Argument / scratch register 4.
r2	a3		Argument / scratch register 3.
r1	a2		Argument / result / scratch register 2.
r0	a1		Argument / result / scratch register 1.

Size of Registers is Architecture Dependent (ARM Cortex-M – 32bits)

ARM Registers and the Stack

- ARM tries to avoid using the stack by using the registers for:
  - Function Input Parameters
  - Local Variables
  - Return Values
- There is a **limited** number of registers
  - Only 12 Registers of 32-bits each

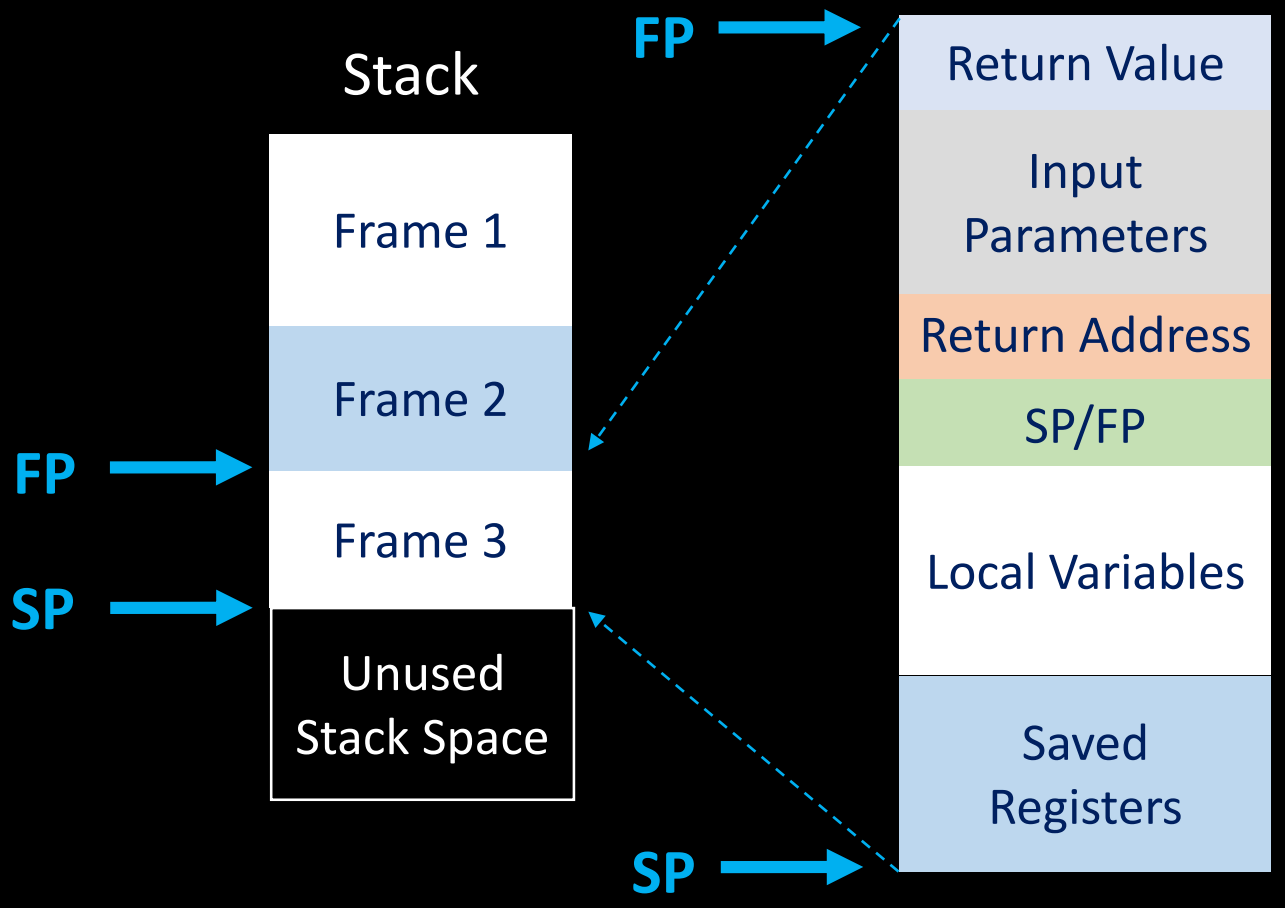
Register	Synonym	Special	Role in the procedure call standard
r15		PC	The Program Counter.
r14		LR	The Link Register.
r13		SP	The Stack Pointer.
r12		IP	The Intra-Procedure-call scratch register.
r11	v8		Variable-register 8.
r10	v7		Variable-register 7.
r9		v6 SB TR	Platform register. The meaning of this register is defined by the platform standard.
r8	v5		Variable-register 5.
r7	v4		Variable register 4.
r6	v3		Variable register 3.
r5	v2		Variable register 2.
r4	v1		Variable register 1.
r3	a4		Argument / scratch register 4.
r2	a3		Argument / scratch register 3.
r1	a2		Argument / result / scratch register 2.
r0	a1		Argument / result / scratch register 1.

Stack Growth

- **Stack Pointer (SP):** Points to end of Used Stack
- **Frame Pointer (FP):** Points to beginning of the stack frame (previous SP)

- Grows Upwards or Downwards

- Similar to data struction Last-In-First-Out Buffer (LIFO)

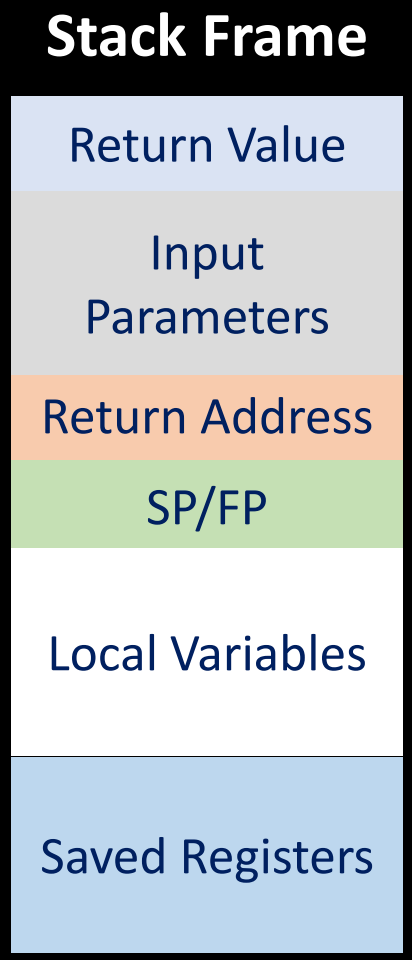


Stack Operations

- Stack Specialized Operatons:
  - Push: Copies data from Registers to Stack
  - Pop: Remove data from Stack to Registers

```
push {r0,r1,r2,lr}
pop {r4,pc}
```

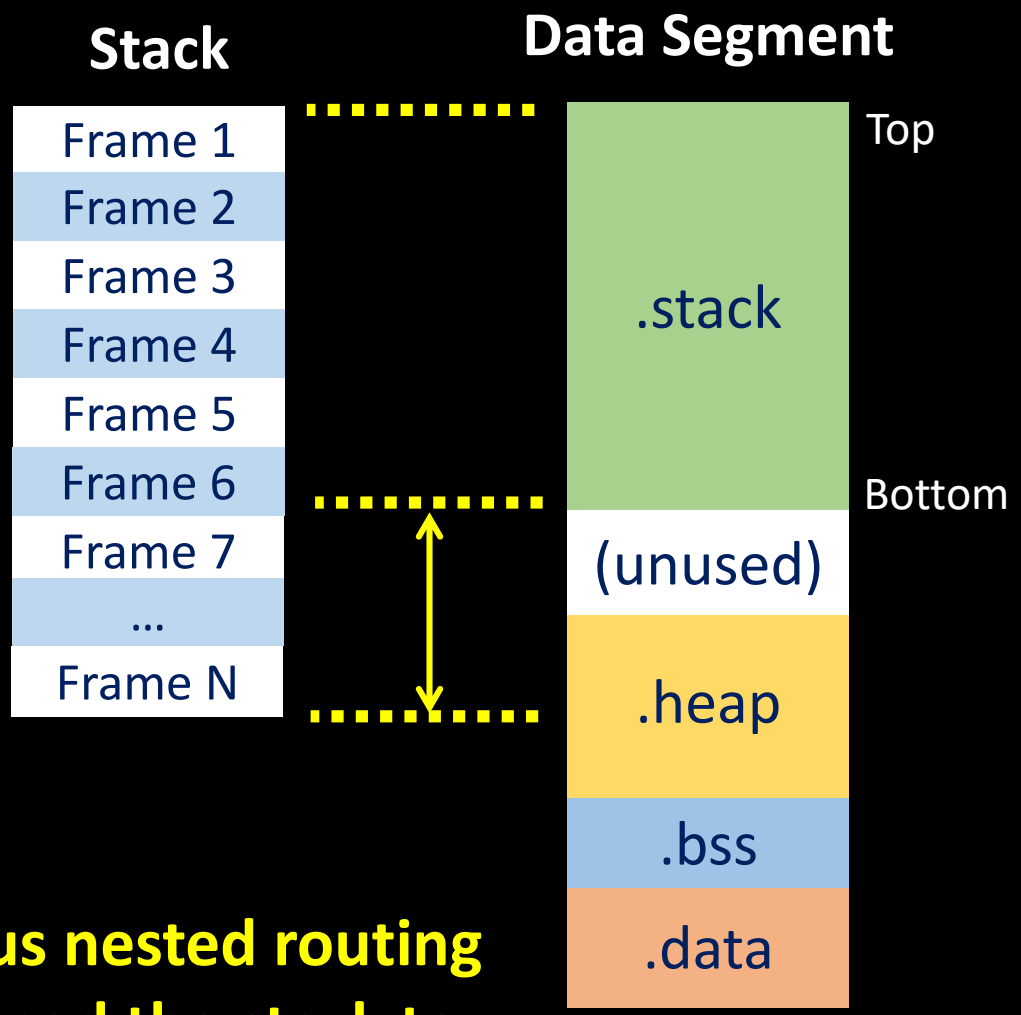
- Can move multiple pieces of data in one instruction
  - Push/Pop
  - STM/LDM – Store/Load multiple Data



Stack Overflow

```c
int main(){
  unsigned int res;
  res = factorial(1000);
  return 0;
}

unsigned int factorial(unsigned int num){
  int ret;
  if (num > 1) {
    ret = num * factorial(num - 1);
  }
  else {
    ret = 1;
  }
  return ret;
}
```

Recursive Function



Numerous nested routing calls caused the stack to overflow into other reserved regions!!!