

L5 functions

Exercises ANS

PUSH/POP Multiple Registers

They are equivalent.

`PUSH {r6, r7, r8}` \longleftrightarrow `PUSH {r8, r7, r6}` \longleftrightarrow `PUSH {r8}`
`PUSH {r7}`
`PUSH {r6}`

They are equivalent.

`POP {r6, r7, r8}` \longleftrightarrow `POP {r8, r7, r6}` \longleftrightarrow `POP {r6}`
`POP {r7}`
`POP {r8}`

- PUSH/POP multiple registers in a single statement: the order in which registers listed in the {register list} does not matter
- When pushing multiple registers, these registers are automatically **sorted by name** and **the lowest-numbered register** is stored to the lowest memory address, i.e. **is stored last**.
- When popping multiple registers, these registers are automatically **sorted by name** and **the lowest-numbered register** is loaded from the lowest memory address, i.e. **is loaded first**.

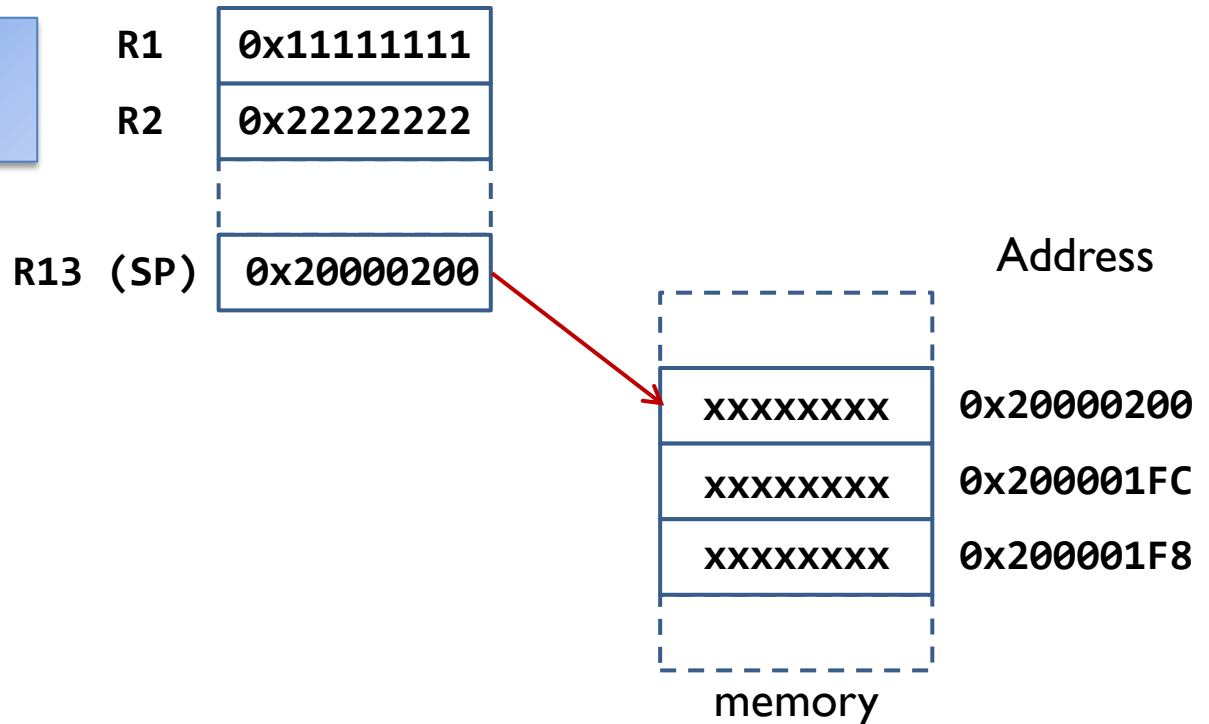
Question: Stack

Before execution

R1==0x11111111

R2==0x22222222

```
PUSH {R1,R2}
POP  {R1}
POP  {R2}
```



▶ Question:

- ▶ What is content of stack, and position of SP, after `PUSH {R1,R2}`?
- ▶ What are the values of R1/R2 after `POP {R2}`?

Answer: Stack

Before execution

R1==0x11111111

R2==0x22222222

PUSH {R1,R2}

POP {R1}

POP {R2}

R1

0x11111111

R2

0x22222222

R13 (SP)

0x20000200

Address

xxxxxxxx

0x20000200

0x22222222

0x200001FC

0x11111111

0x200001F8

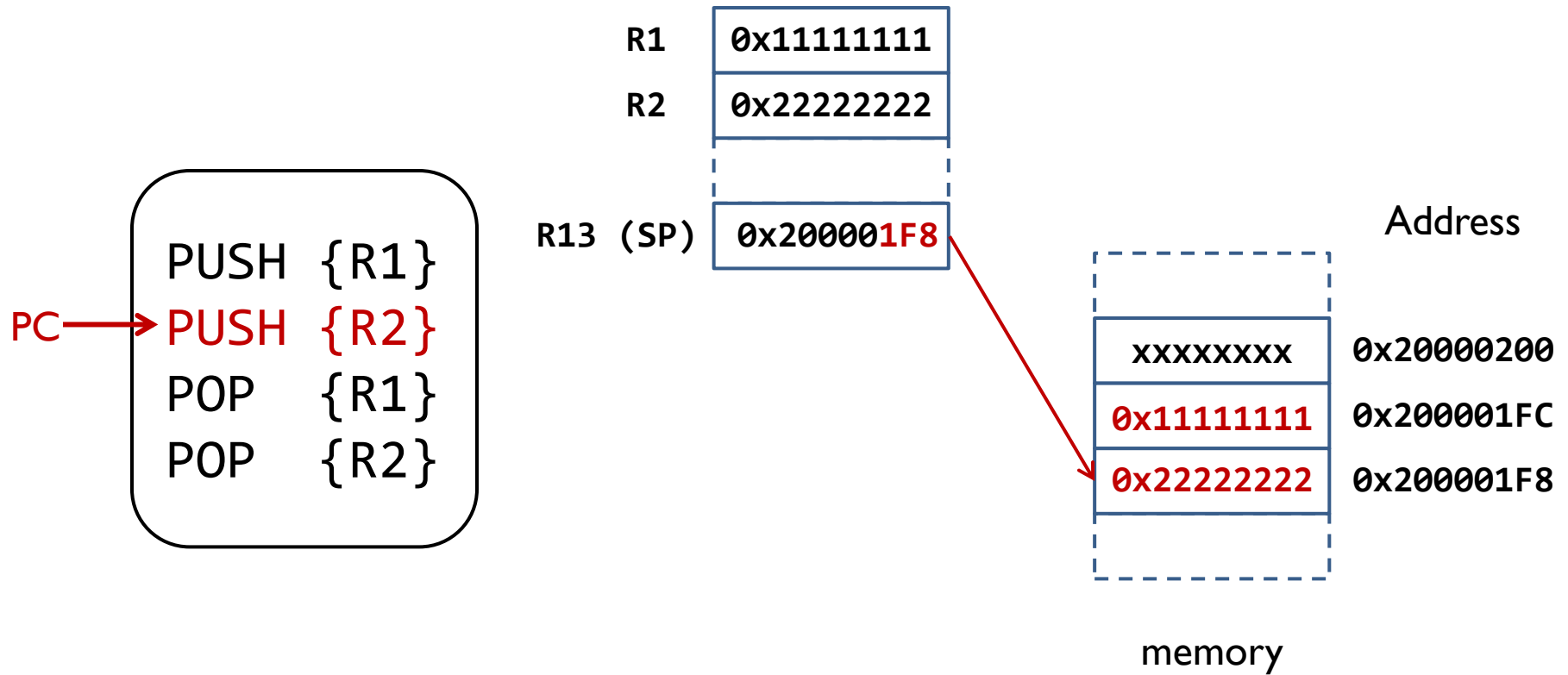
memory

► Answer:

► Shown in figure

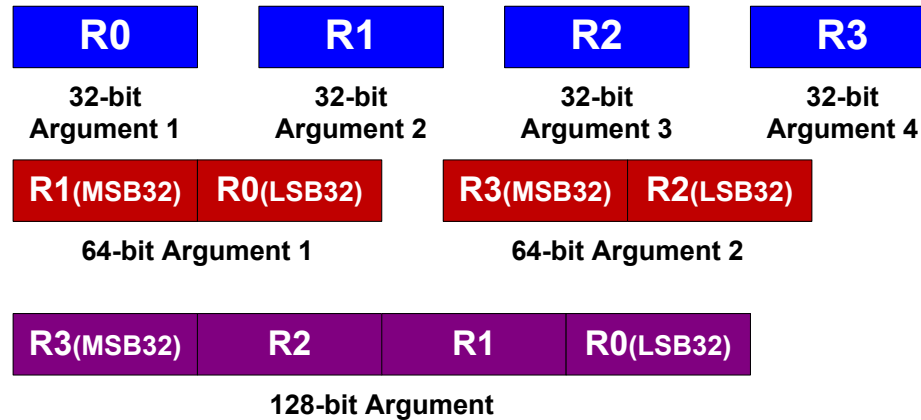
► After POP {R2}, R1==0x11111111, R2==0x22222222

Example: Swap R1 & R2

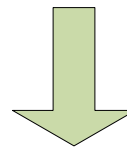
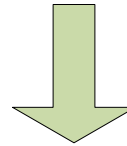


Review

Passing Arguments via Registers R0-R3



Extra arguments are pushed to the stack by the caller. The caller is responsible to pop them out of the stack after the subroutine returns.



Review

Additional Arguments Passed on Stack

Registers

R0	R1	R2	R3
----	----	----	----

Stack in Memory

--	--	--	--	--	--	--

```
foo (int i0, int i1, int i2, int i3)
```

Registers

i0	i1	i2	i3
----	----	----	----

Stack in Memory

--	--	--	--	--	--	--

```
foo (int i0, char a1, double D)
```

Registers

i0	a1	D
----	----	---

Stack in Memory

--	--	--	--	--	--	--

```
foo (int i0, int i1, double D, int i2, int i3)
```

Registers

i0	i1	D
----	----	---

Stack in Memory

i2	i3					
----	----	--	--	--	--	--

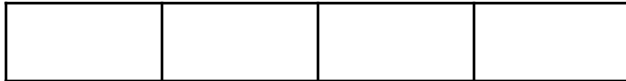
Caller passes arguments i0, i1, D in registers R0-R3 directly; pushes additional arguments i2 and i3 onto the stack before function call (details not covered in this lecture)

Question: Argument Passing

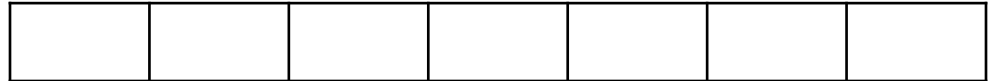
- ▶ Which registers are used to pass the arguments and return the result?

long fun (short a1, char a2, double a3, int a4, char a5)

Registers



Stack in Memory



Answer: Argument Passing

- ▶ Which registers are used to pass the arguments and return the result?

```
long fun (short a1, char a2, double a3, int a4, char a5)
```

Registers

a1	a2	a3
----	----	----

Stack in Memory

a4	a5					
----	----	--	--	--	--	--

- ▶ Each argument of 8-bit char, or 16-bit short, is passed in 1 32-bit register; cannot use 1 register to pass more than 1 arguments