



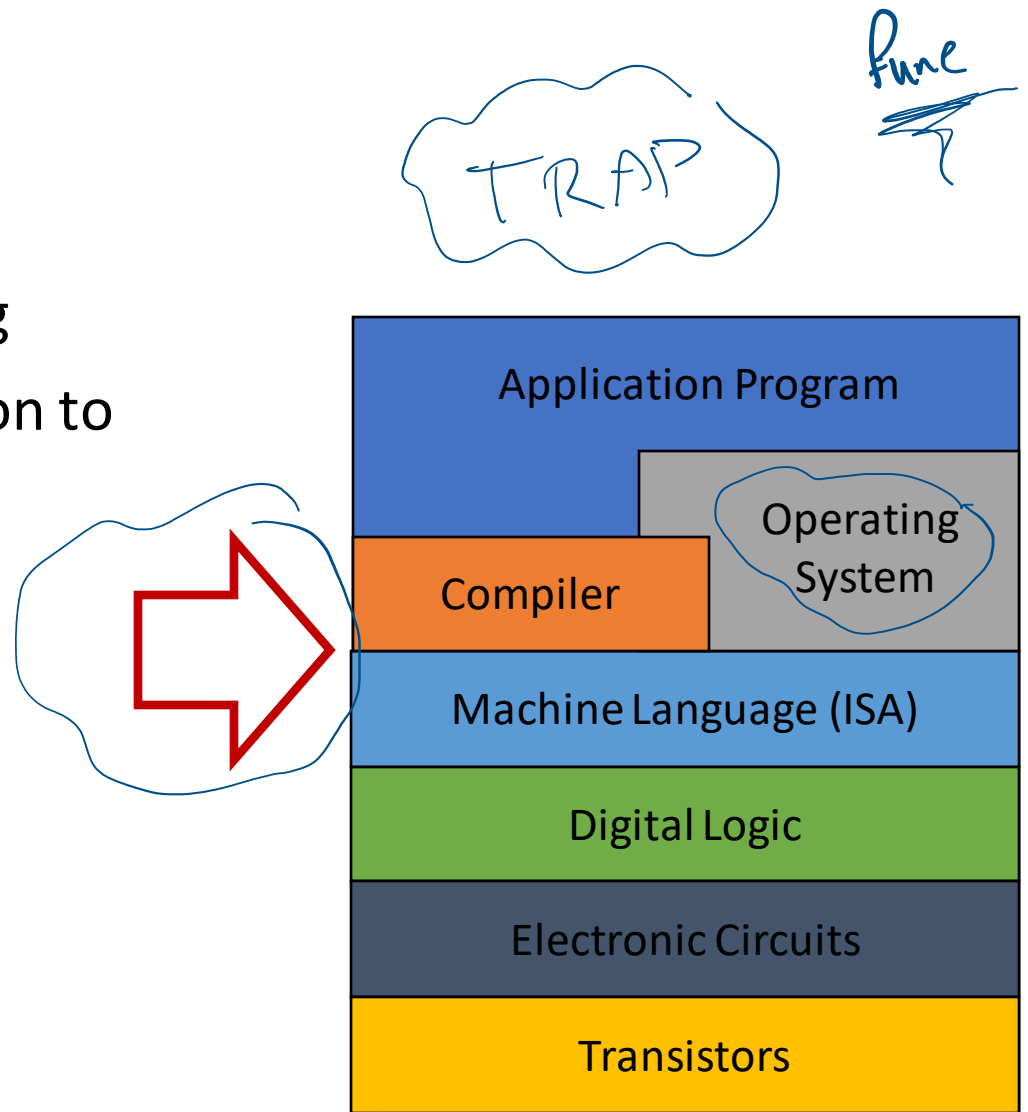
Subroutine

EECS388 Fall 2022

© Prof. Mohammad Alian

Context

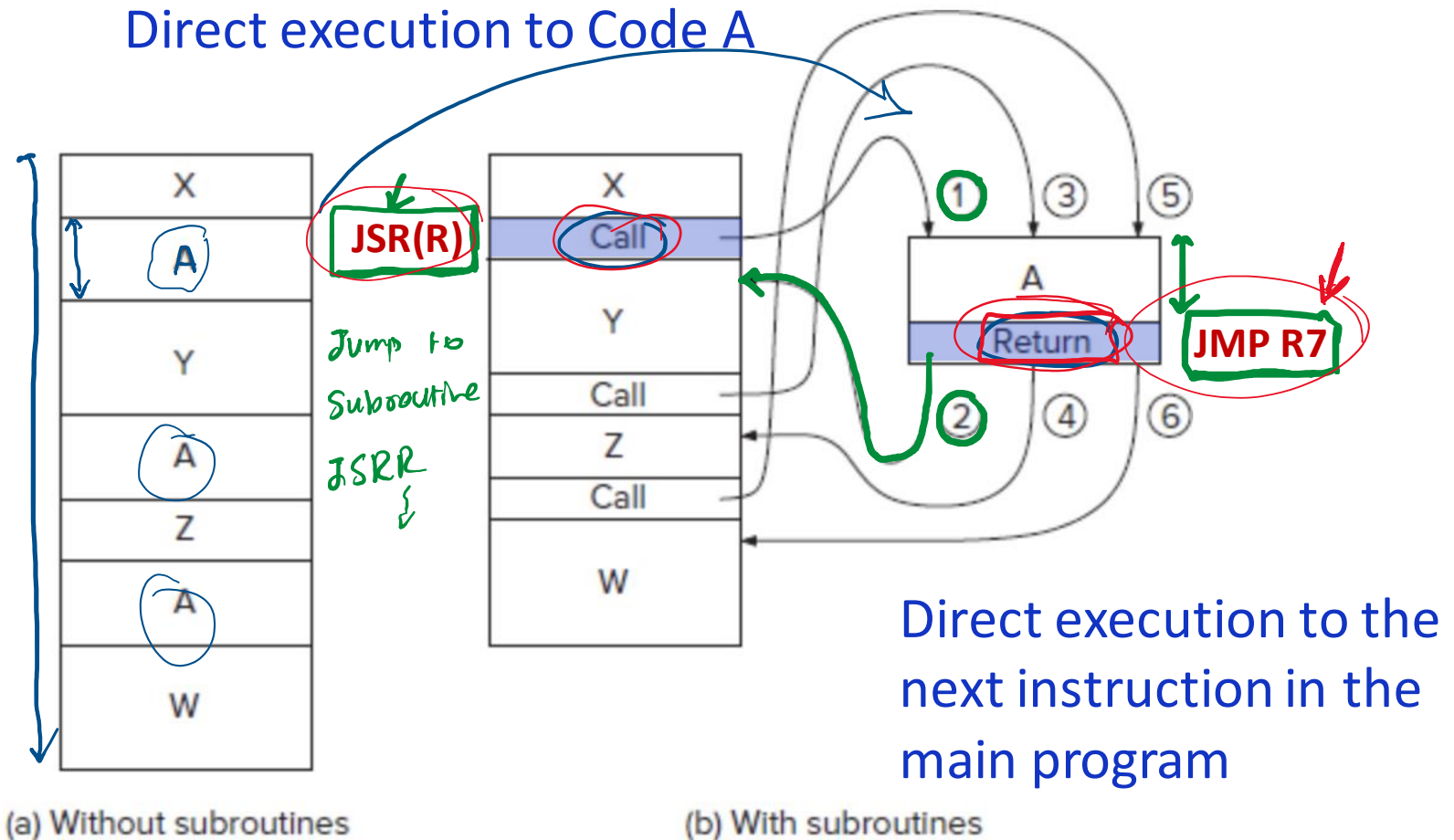
- Recommended reading
Chapter 8 of “Introduction to Computing,” Patt, Patel



We cannot write a complex application in one function

- Some parts of the code need to be used many times
- Solution: *functions* or *subroutines*
 - Also, Enables us to share codes and have libraries ...

The Call/Return Mechanism



(Review) JSR, JSRR Instructions

JSR LABEL

JSRR BaseR

- Operation:

Temp = PC

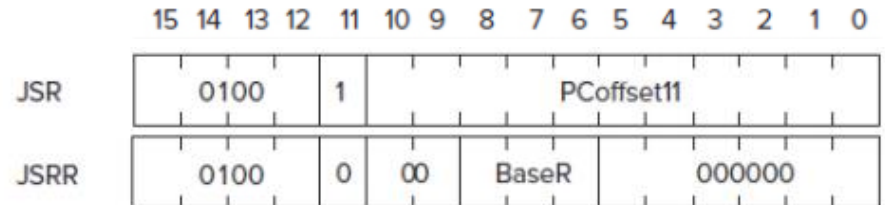
if (bit[11] == 0)

PC = BaseR

else

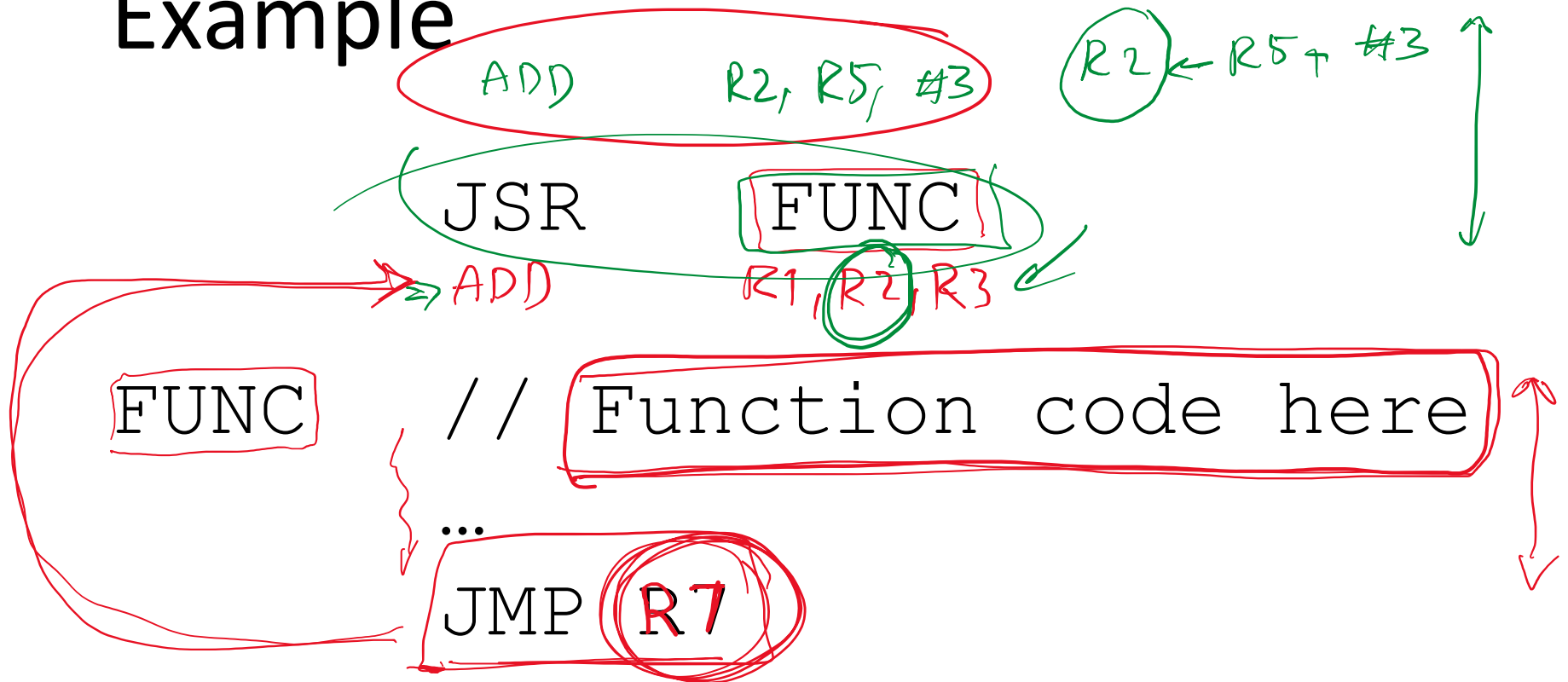
PC = PC + SEXT(PCoffset11)

R7 = TEMP



CALL

Example



ALERT!

- Writing into a register means that we lose the previous value!
- We need to save registers
 - If the value will be destroyed by subsequent instructions
 - If we need it after subsequent instructions
- This can be a problem with subroutines!

Solution: Save and Restore Registers to/from Memory

// R1 and R2 are used

...

ST

ST

R1, SaveR1

R2, SaveR2

} save

⇒ JSR

FUNC

RT ← PC

←

// R1 and R2 values are reused

...

FUNC

// Function code here

save R7

R7

...

→ LD

LD

R1, SaveR1

R2, SaveR2

} restore

JMP

R7

Return

RET

restore

SaveR1

.BLKW

1

SaveR2

.BLKW

1

Solution: Save and Restore Registers to/from Memory

~~// R1 and R2 are used~~ *all registers are use*

...

ST R1, SaveR1

ST R2, SaveR2

JSR FUNC

ID R7, SaveR7 // R1 and R2 values are reused

... *all registers*

FUNC // Function code here

...

only use R3

ID R3, SaveR3

JMP R7

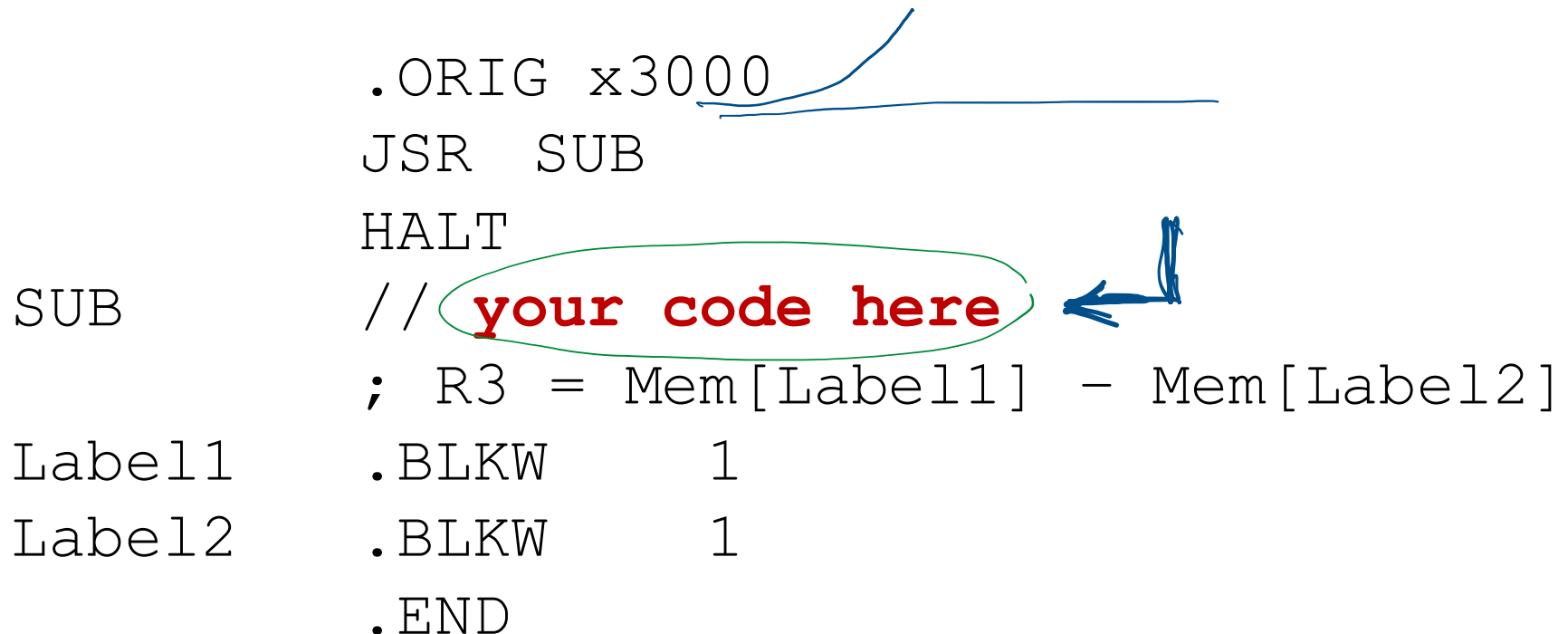
SaveR~~1~~₃ .BLKW 1

SaveR~~2~~₇ .BLKW 1

Example

Implement a subroutine that subtract the value of memory stored in Label1 and Label2 into R3

```
.ORIG x3000
JSR SUB
HALT
SUB // your code here
; R3 = Mem[Label1] - Mem[Label2]
Label1 .BLKW 1
Label2 .BLKW 1
.END
```



$R3 \leftarrow \text{Mem}[\text{label1}] - \text{Mem}[\text{label2}]$

{ ST R1, LR1 R1
ST R2, LR2
ST R3, LR3

R2

JSR SUB
LD R1, LR1
LD R2, LR2
LD R3, LR3

SUB

LD R1, Label1

LD R2, Label2

NOT R2, R2

ADD R2, R2, #1

ADD R3, R1, R2