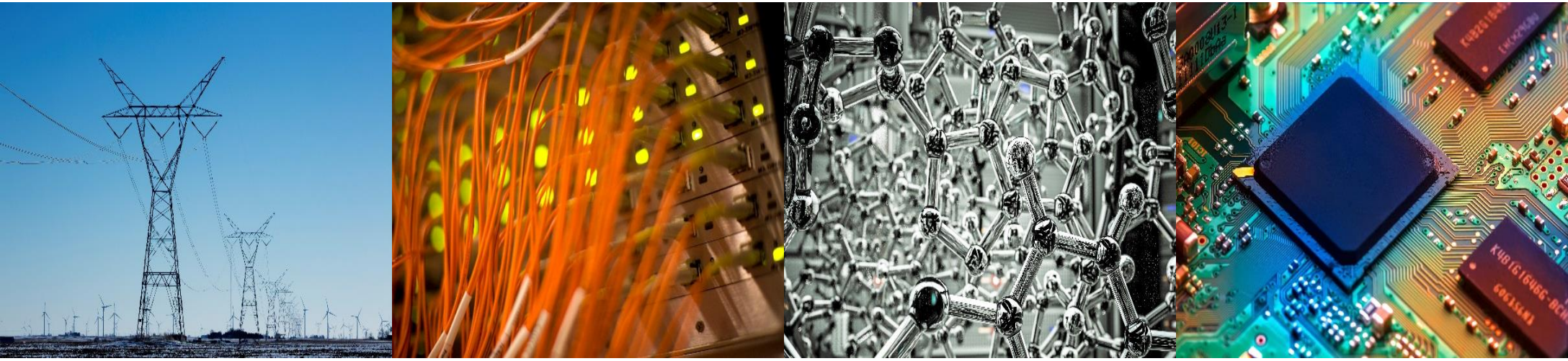
ECE 220 Computer Systems & Programming

Lecture 2 – Repeated Code: TRAPs and Subroutines

August 29, 2024



- **Mock quiz (extra-credit) is now available for reservation on PrairieTest and should be taken between 9/9 and 9/11 at CBTF**

ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING

Repeated Code

Example: input from keyboard

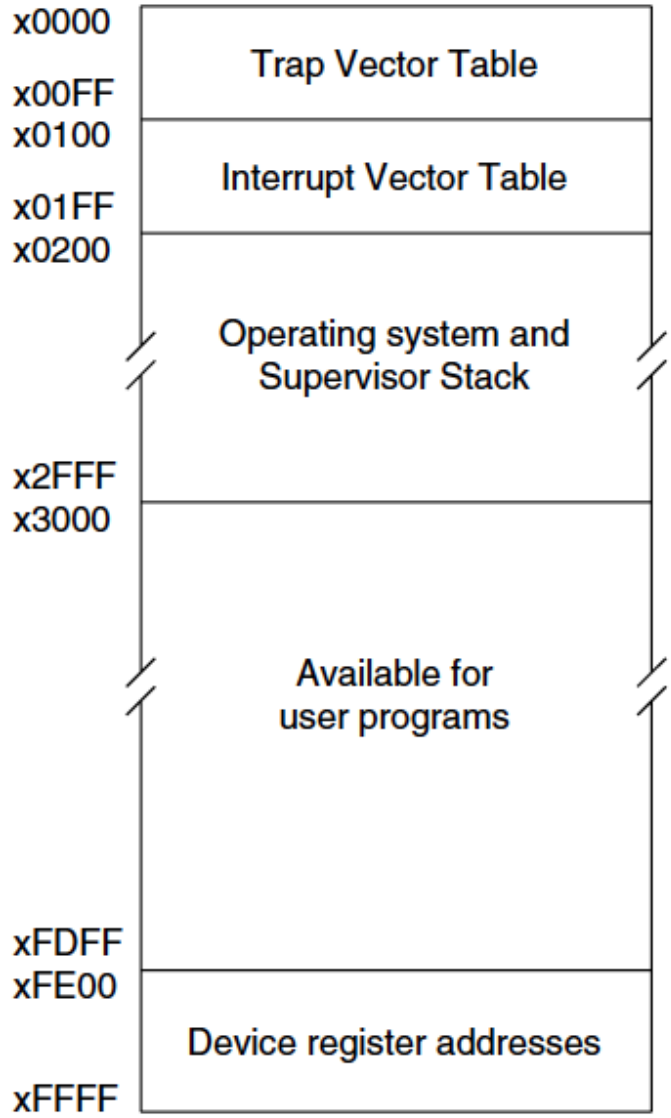
- It's used often and has too many specific details for most programmers.
- Improper usage could breach security of the system.

Solution: make this part of the OS

Service Call / System Call

1. User program invokes system call
2. Operating system code performs operation
3. Returns control to user program

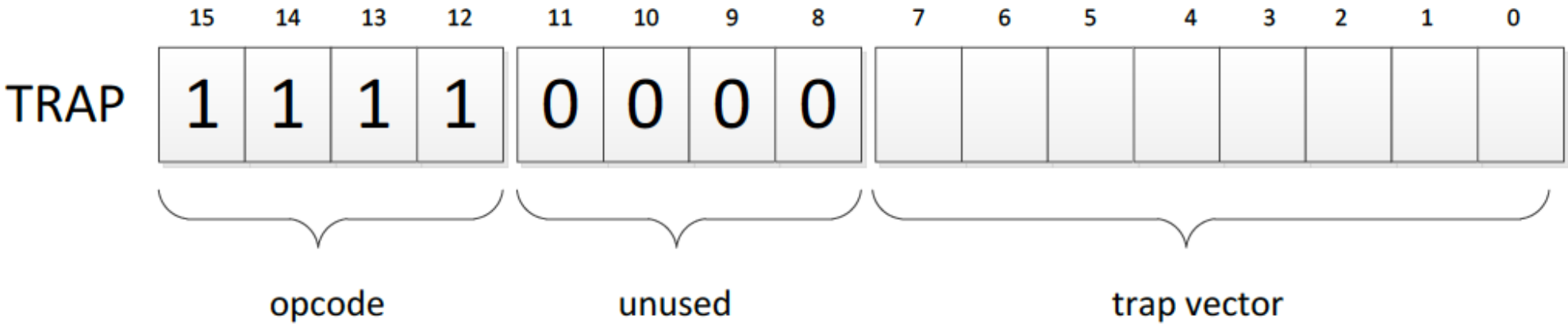
In LC-3, this is done through the TRAP mechanism.



TRAP Mechanism

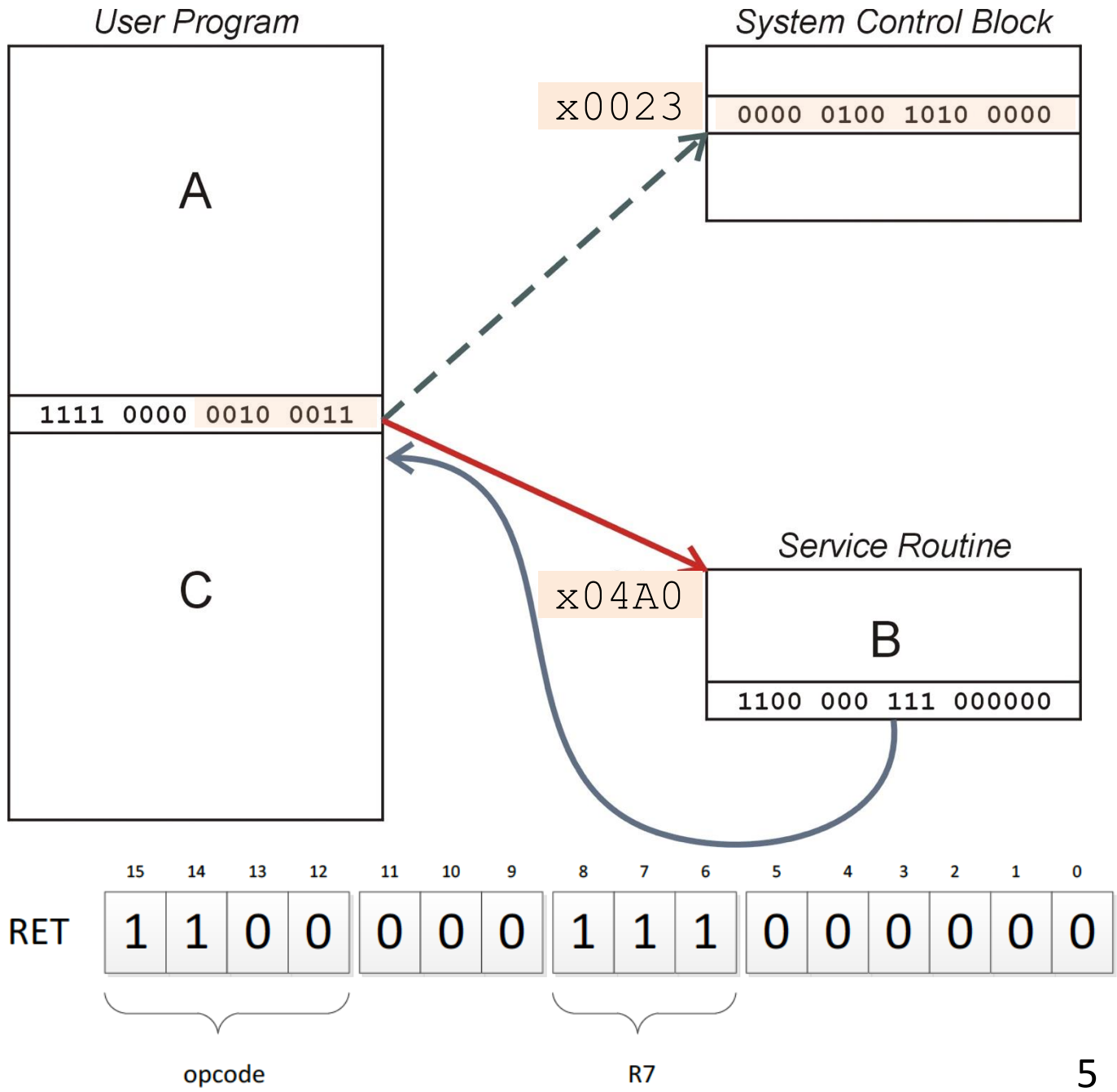
1. A set of service routines executed by the OS
2. A table of the starting addresses of these service routines
3. The TRAP instruction
4. A Linkage back to the user program

TRAP Instruction in LC-3



Vector	Symbol	Routine
x20	GETC	Read a single character (no echo)
x21	OUT	Output a character to monitor
x22	PUTS	Write a string to monitor
x23	IN	Print prompt to monitor, read and echo character from keyboard
x24	PUTSP	Write a string to monitor, two characters per memory location
x25	HALT	Halt the program
x26		Write a number to monitor (undocumented)

Flow of Control



TRAP Example

```
.ORIG x3000

AND R0, R0, #0 ;init R0
ADD R0, R0, #4 ;set R0 to 4
ADD R7, R0, #5 ;set R7 to 9
ADD R0, R0, #1 ;increment R0
ADD R7, R7, #1 ;increment R7

IN             ;same as 'TRAP x23'

ADD R0, R0, #1 ;increment R0
ADD R7, R7, #1 ;increment R7

HALT
.END
```

- What are the values in R0 and R7 right before IN? How about right before HALT?

Saving & Restoring Registers

We must save the value of a register if its value will be destroyed by the service routine and the value will be needed after that action.

Two Conventions for Saving & Restoring Registers

1. Callee-saved (knows what it alters, but does not know what will be needed by calling routine)

- Before start, _____
- Before return, _____

2. Caller-saved (know what it needs later, but may not know what gets altered by callee routine)

-
-

Subroutines

Service routines (TRAP) provide 3 main functions

- Shield programmers from system-specific details
- Write frequently-used code just once
- Protect system recourses from malicious/clumsy programmers

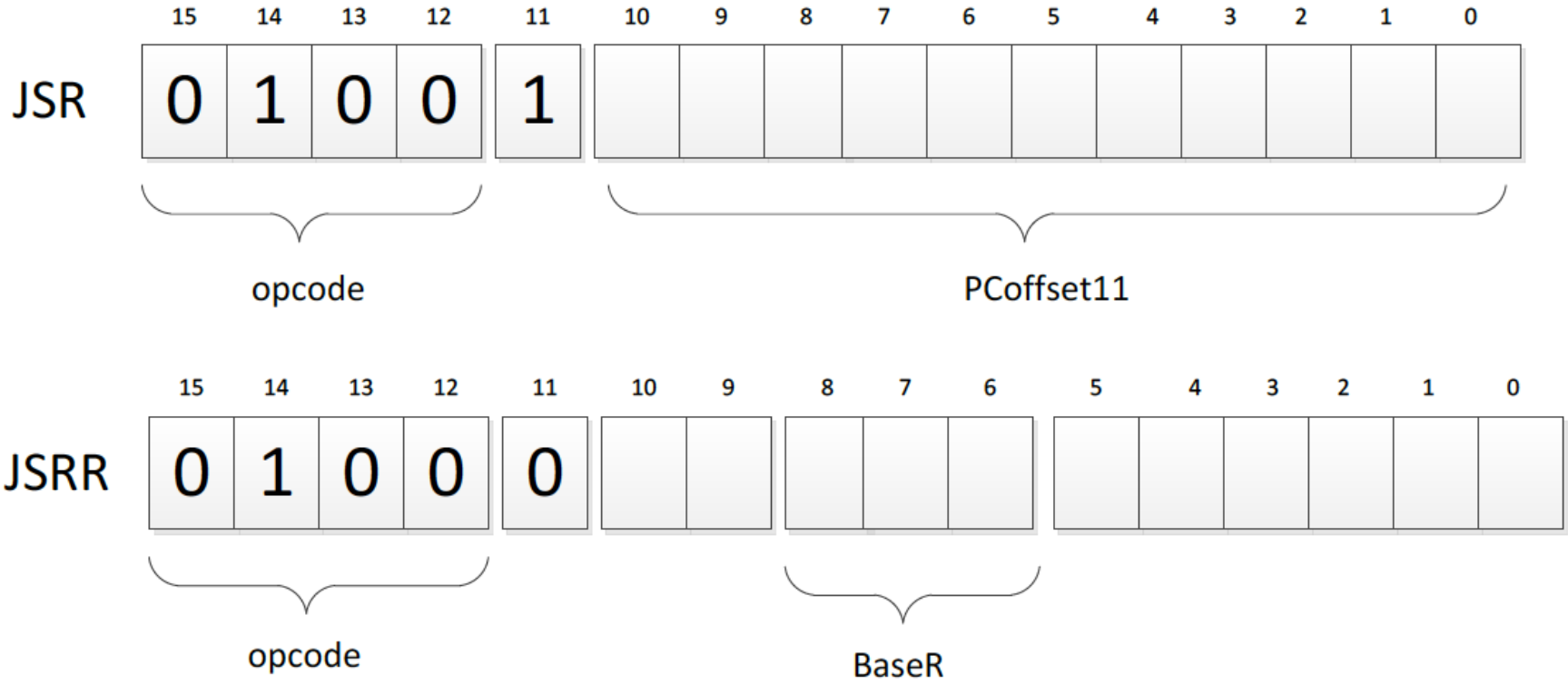
Subroutines provide the same functions for non-system (user) code

-
-
-
-

➢ **What are some of the reasons to use subroutines?**

Invoking Subroutines using JSR/JSRR

- A subroutine in LC-3 can be invoked by using JSR/JSRR instruction



- To return from a subroutine, use **RET** instruction

Passing Information to/from Subroutines

Argument

- A value passed into a subroutine (needed by the subroutine to do its job)

Return Value

- A value passed out of a subroutine (the value you called the subroutine to compute)

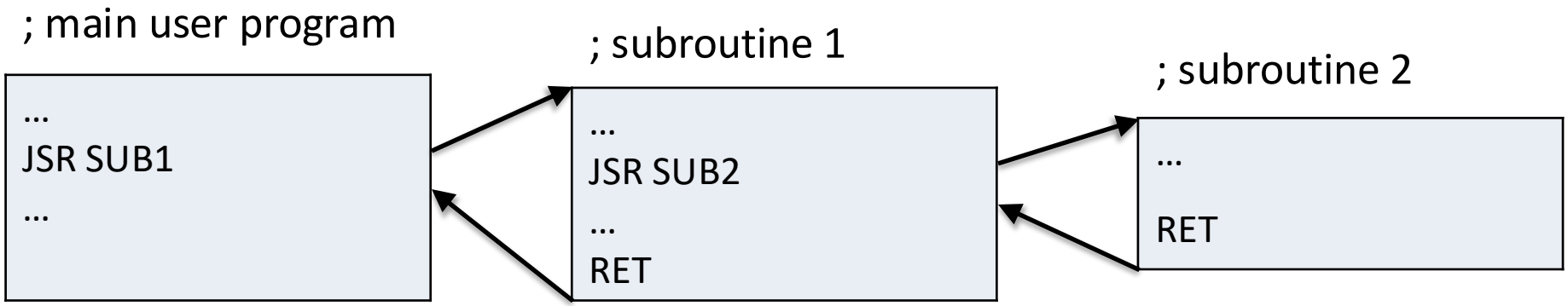
Saving/Restoring Registers in Subroutines

1. Generally use _____ , except for _____

2. Save *anything* that the subroutine will *alter internally* that should not be visible when the subroutine returns

3. It's good practice to restore _____

- **Nested subroutines**



Subroutine Example

```
.ORIG x3000
...
JSR SUBTR
...
HALT

;SUBTR subroutine computes R1 minus R2
;IN: R1, R2
;OUT: R0 ← R1-R2
SUBTR
      NOT R2, R2
      ADD R7, R2, #1       ;get -R2
      ADD R0, R1, R7       ;R0 = R1-R2
      RET


.END
```

- Is there any bugs?
- Where in the code should we save and restore registers?
- How can we compute $2x^2 - 3x + 1$?