ECE 220 Computer Systems & Programming

Lecture 4 – TRAPs (2nd Edition) and Subroutines

Yih-Chun Hu
adapted from material by Profs. Yuting Chen, Sanjay Patel, Volodymyr Kindratenko

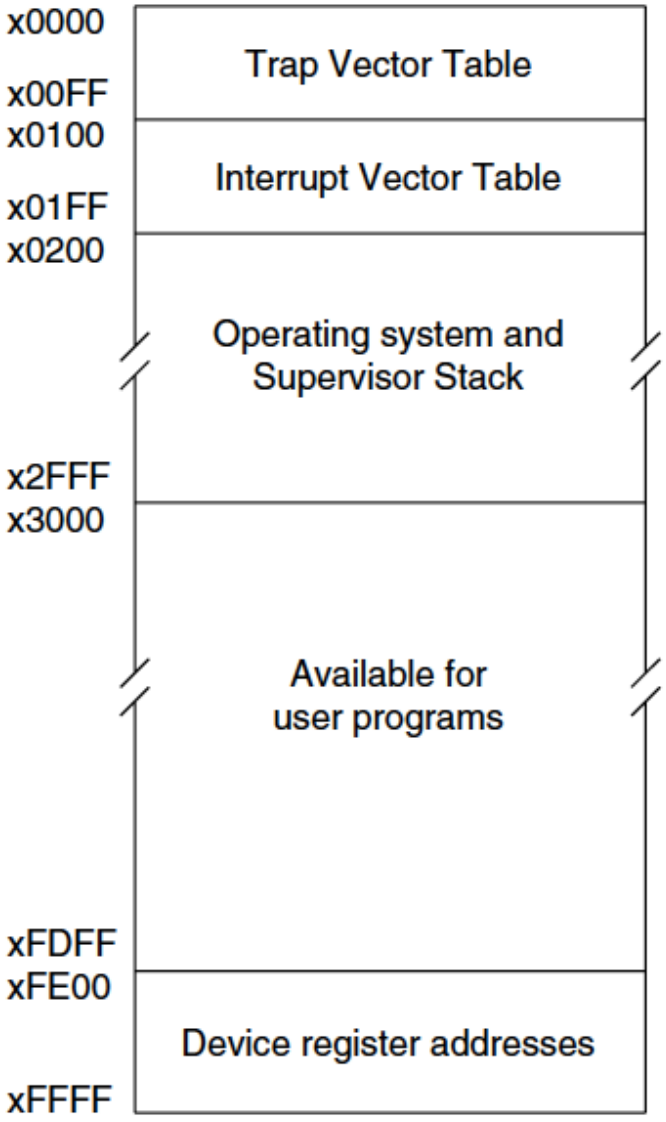
ECE ILLINOIS

ILLINOIS

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 _____ mechanism.



TRAP Mechanism

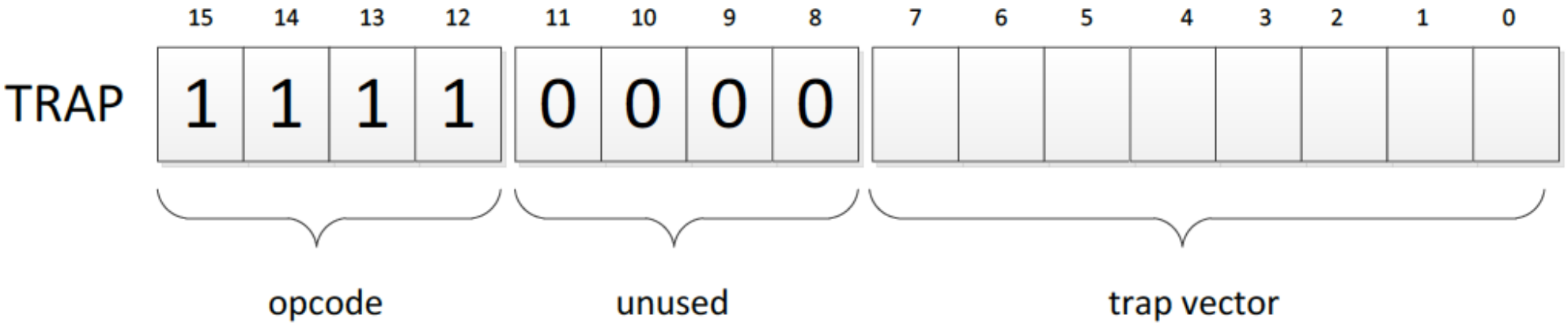
1. A set of service routines executed by the OS

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



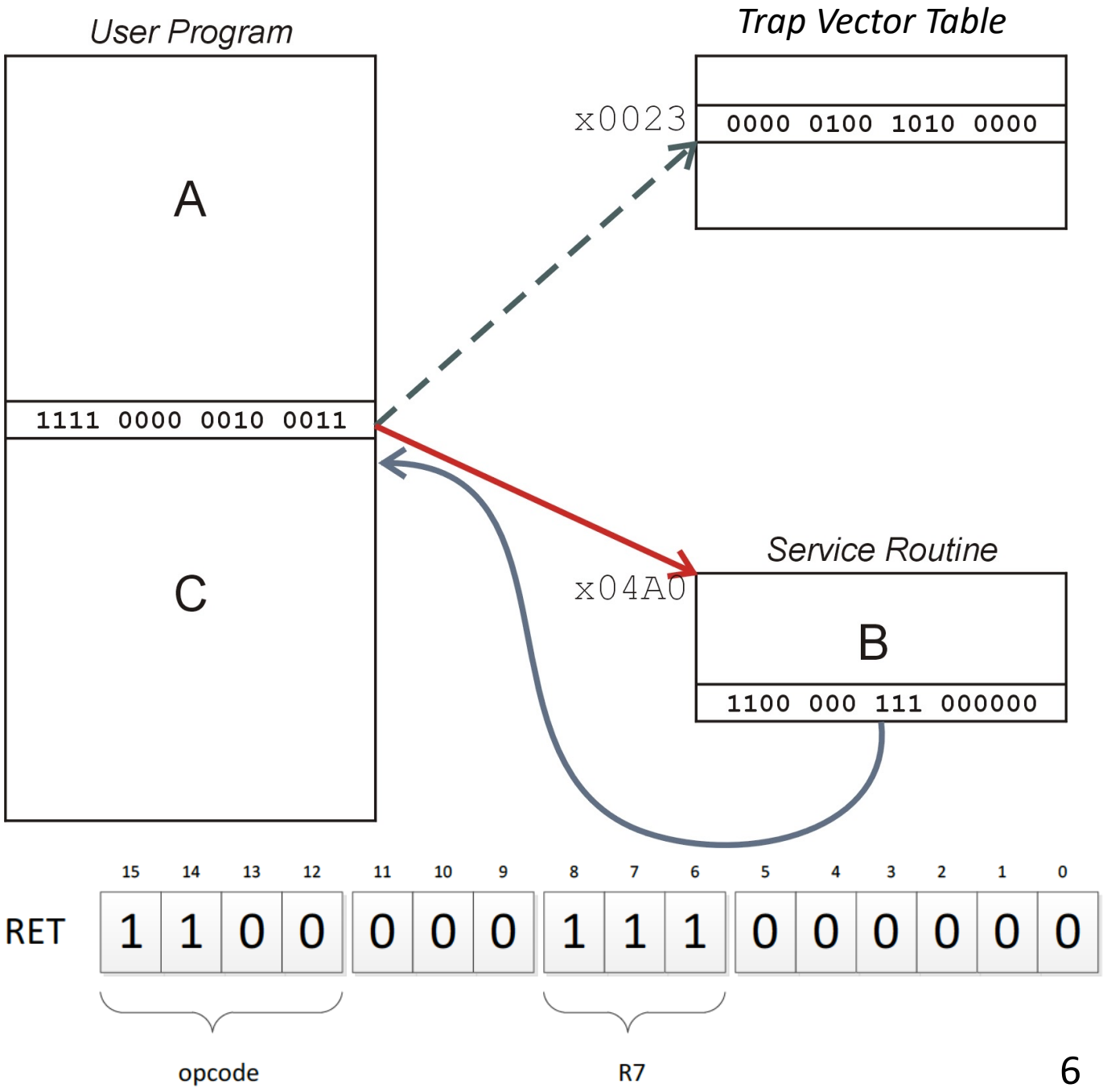
vector	*symbol*	*routine*
x20	**GETC**	read a single character (no echo)
x21	**OUT**	output a character to the monitor
x22	**PUTS**	write a string to the console
x23	**IN**	print prompt to console, read and echo character from keyboard
X23	**PUTSP**	write a string to the console; two chars per memory location
x25	**HALT**	halt the program
x26		write a number to the console (undocumented)

Flow of Control

TRAP RTL (2nd ed)

R7 ←PC

PC ← M[ZEXT[trapvec8]]



TRAP Example

```
.ORIG x3000

AND R0, R0, #0
ADD R0, R0, #5    ;init R0 and set it to 5
ADD R7, R0, #2    ;set R7 to 7

IN                ;same as 'TRAP x23'

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

HALT
.END
```

- Question: What are the values in R0 and R7 before HALT?

Saving & Restoring Registers

We must save the value of a register if

- Its value will be destroyed by the service routine
- We will need to use the value after that action

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

- Before start, save the registers that the callee uses
- Before return, restore those registers to their original values

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

- Before call, save registers that the caller needs
- After return, restore those registers

Subroutines

Service routines (TRAP) provide 3 main functions

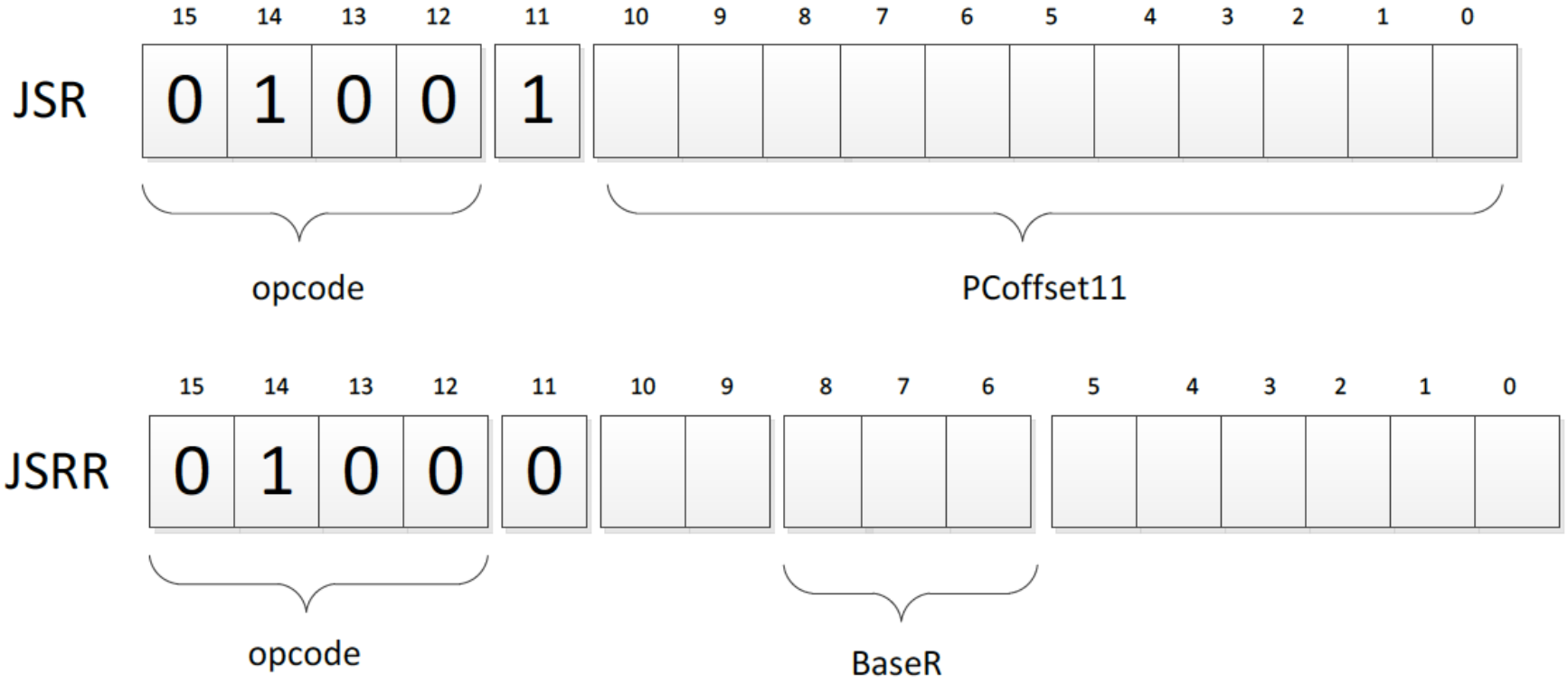
- **Shield programmers from system-specific details**
- **Write frequently-used code just once**
- **Protect system recourses from malicious/clumsy programmers**

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

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

JSR/JSRR

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



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