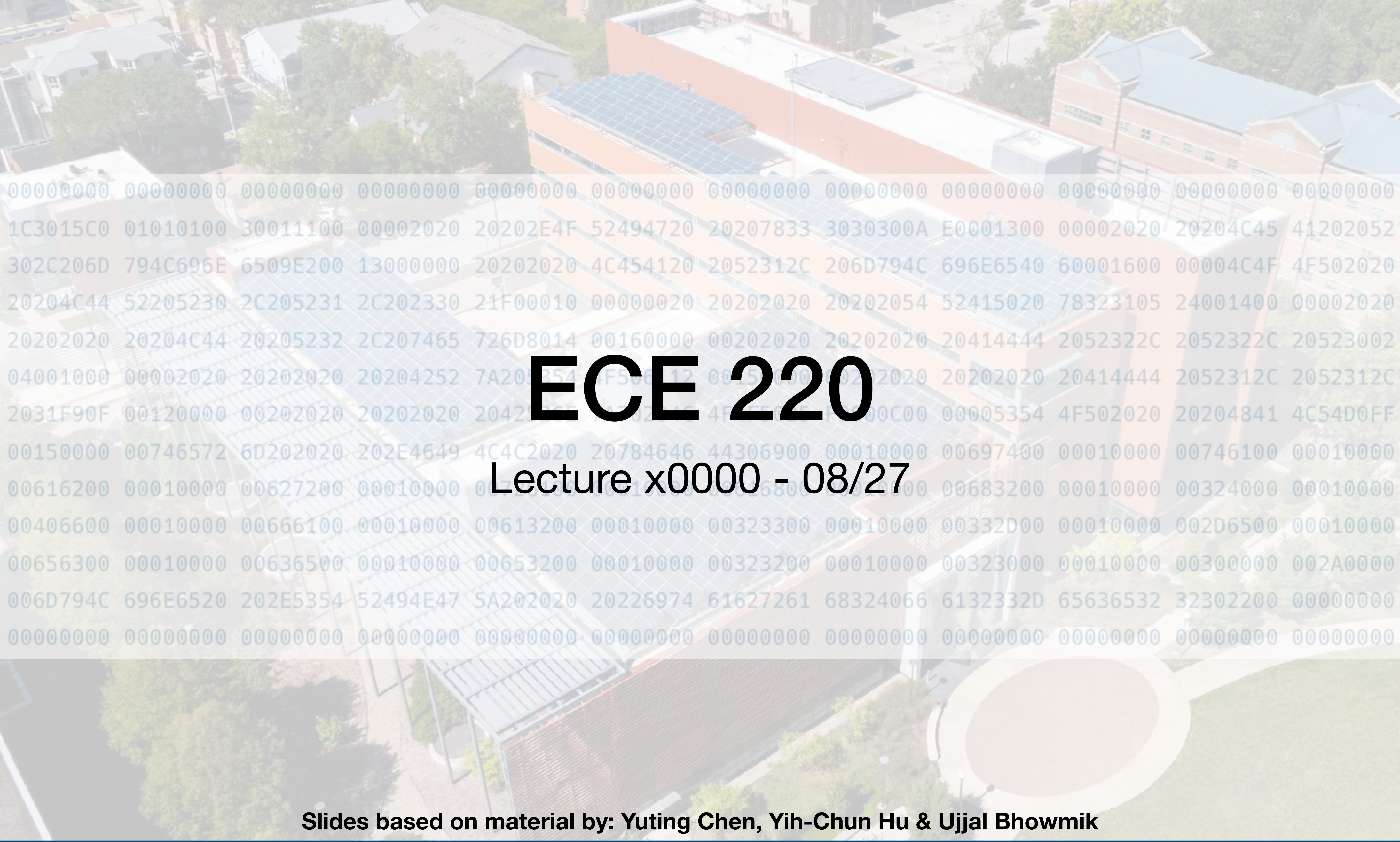


ECE 220

Lecture x0000 - 08/27

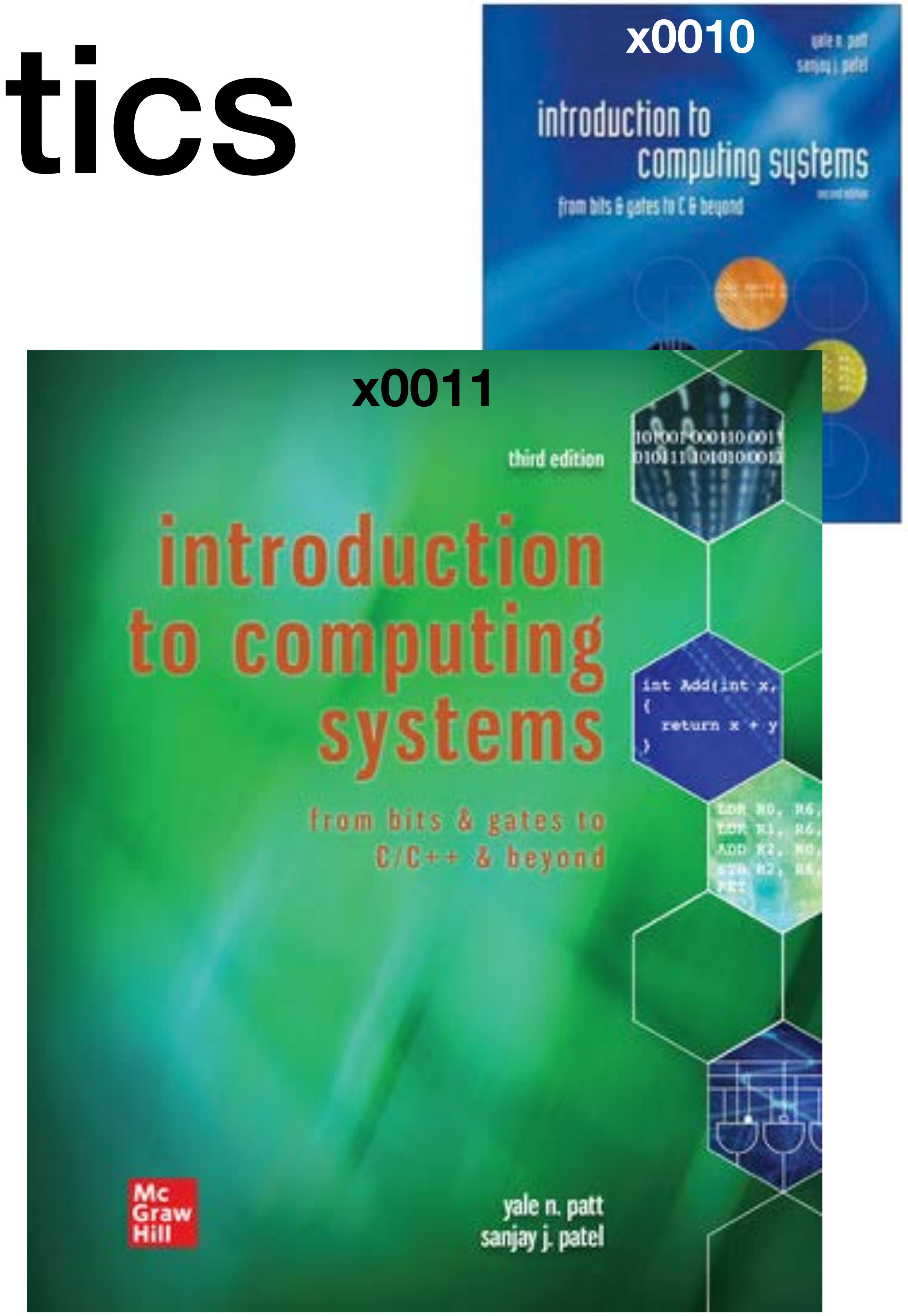
Slides based on material by: Yuting Chen, Yih-Chun Hu & Ujjal Bhowmik

Course logistics

- Lectures: Tuesdays & Thursday
  - Three sections offered by different instructors
  - Prof. Yuting Chen (1230, BL1), Prof. Yih-Chun Hu (1100, BL2) and **this one** (1400, BL3).
- Labs: Fridays
  - Starts on the hour, every hour from 0900 hrs until 1650 hrs
- Office hours: Schedule posted to website

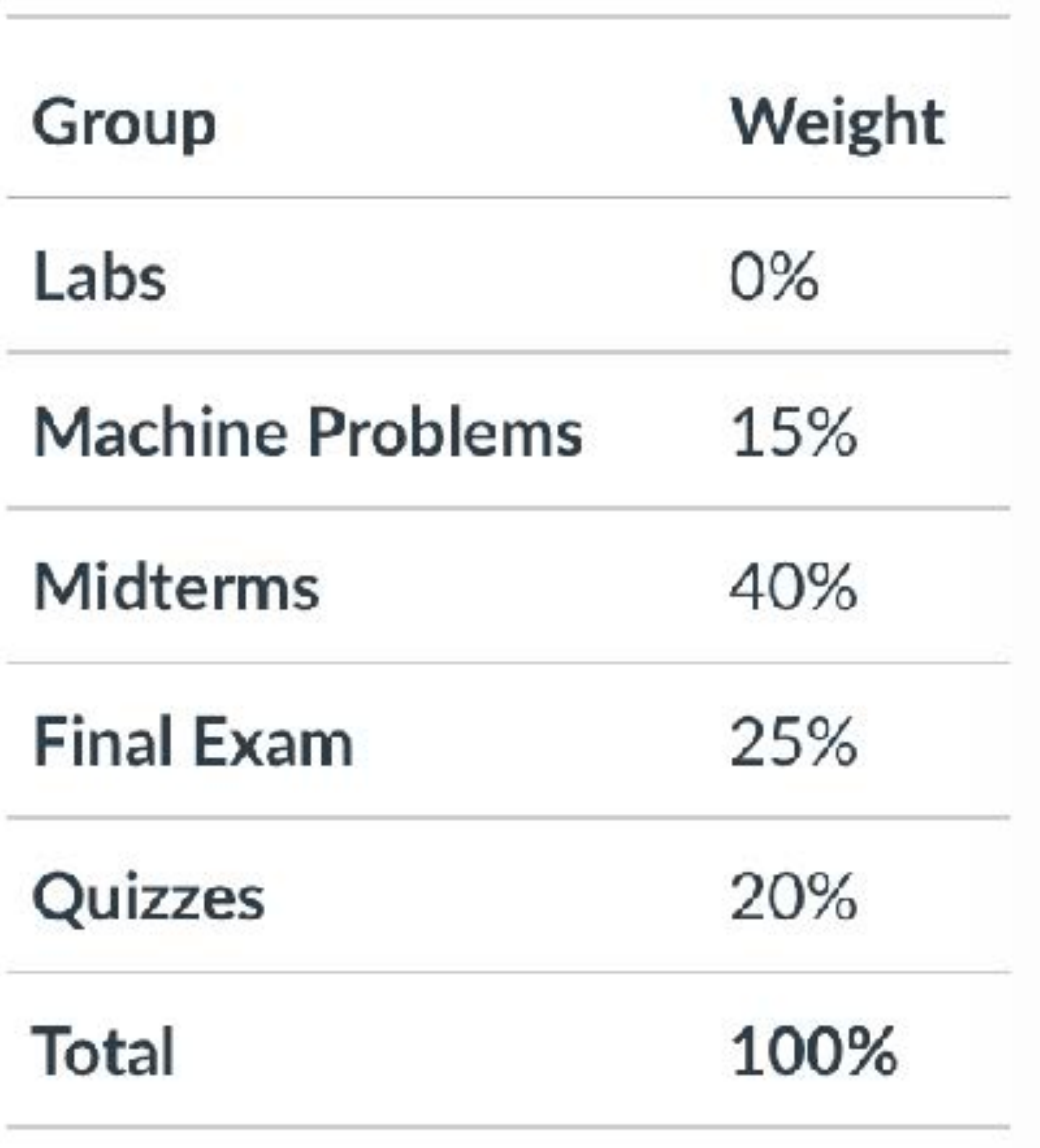
Course logistics

- Course Website (and syllabus)
- Grading: Gradescope + autograder
- Discussions: Campuswire
- Quizzes: CBTF
- Machine problems (MPs): Github
- Textbook: Patt & Patel (3rd Ed)


x0010
introduction to computing systems
from bits & gates to C & beyond
yale n. patt
sanjay j. patel
x0011
third edition
introduction to computing systems
from bits & gates to C/C++ & beyond
Mc Graw Hill
yale n. patt
sanjay j. patel

Course logistics

- MPs: 12 in total, lowest dropped (except MP 12)
- Quizzes (in-person in CBTF): 6 total, lowest dropped
- Exams (in-person, on-paper): 09/26 and 10/31
- Labs: make up points lost on MPs



Group	Weight
Labs	0%
Machine Problems	15%
Midterms	40%
Final Exam	25%
Quizzes	20%
Total	**100%**

Syllabus

Quick recap of ECE 120

Computation

Von Neumann model

- Five major components:
  1. Memory
  2. Input
  3. Output
  4. Processing unit
  5. Control unit

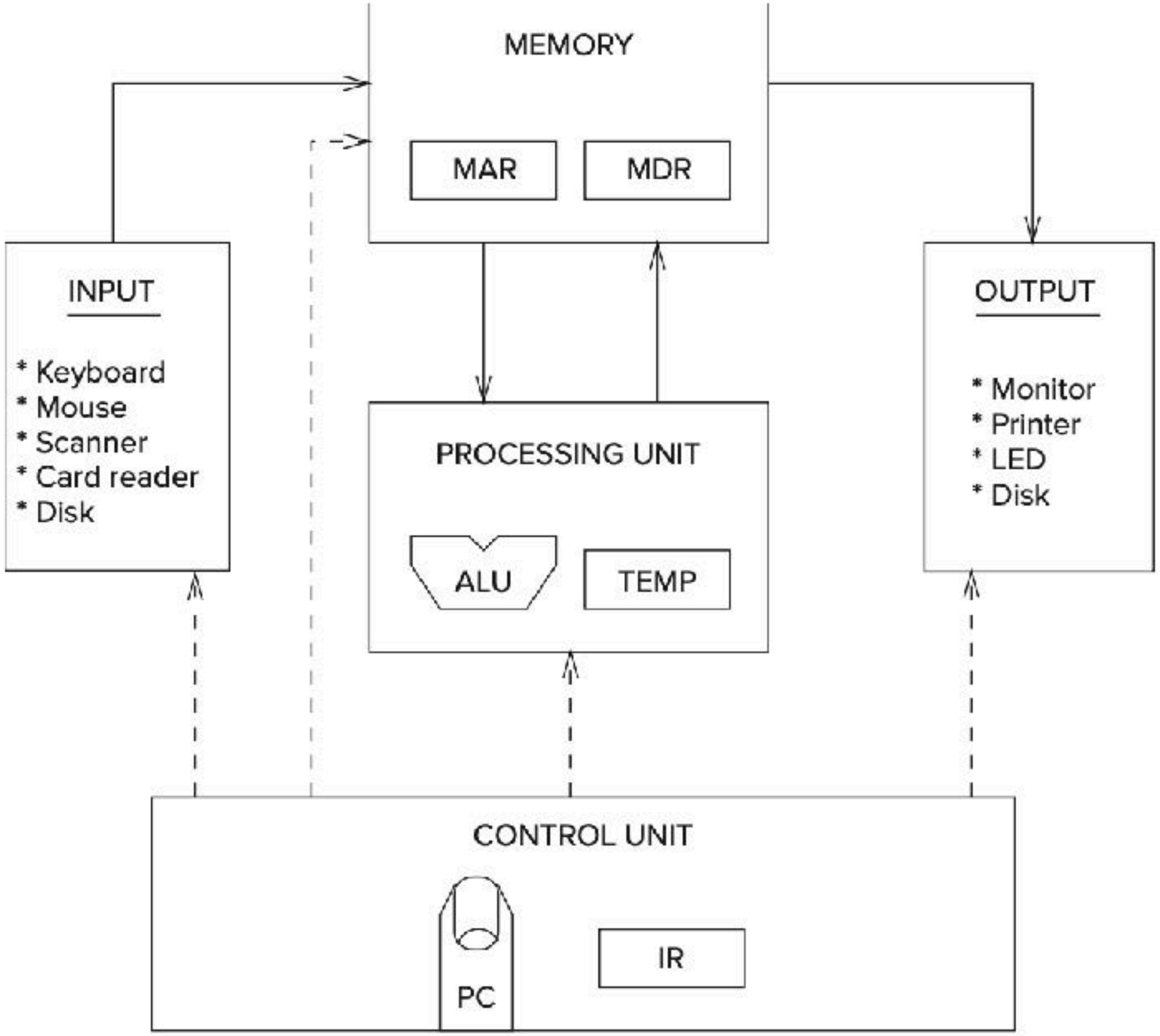


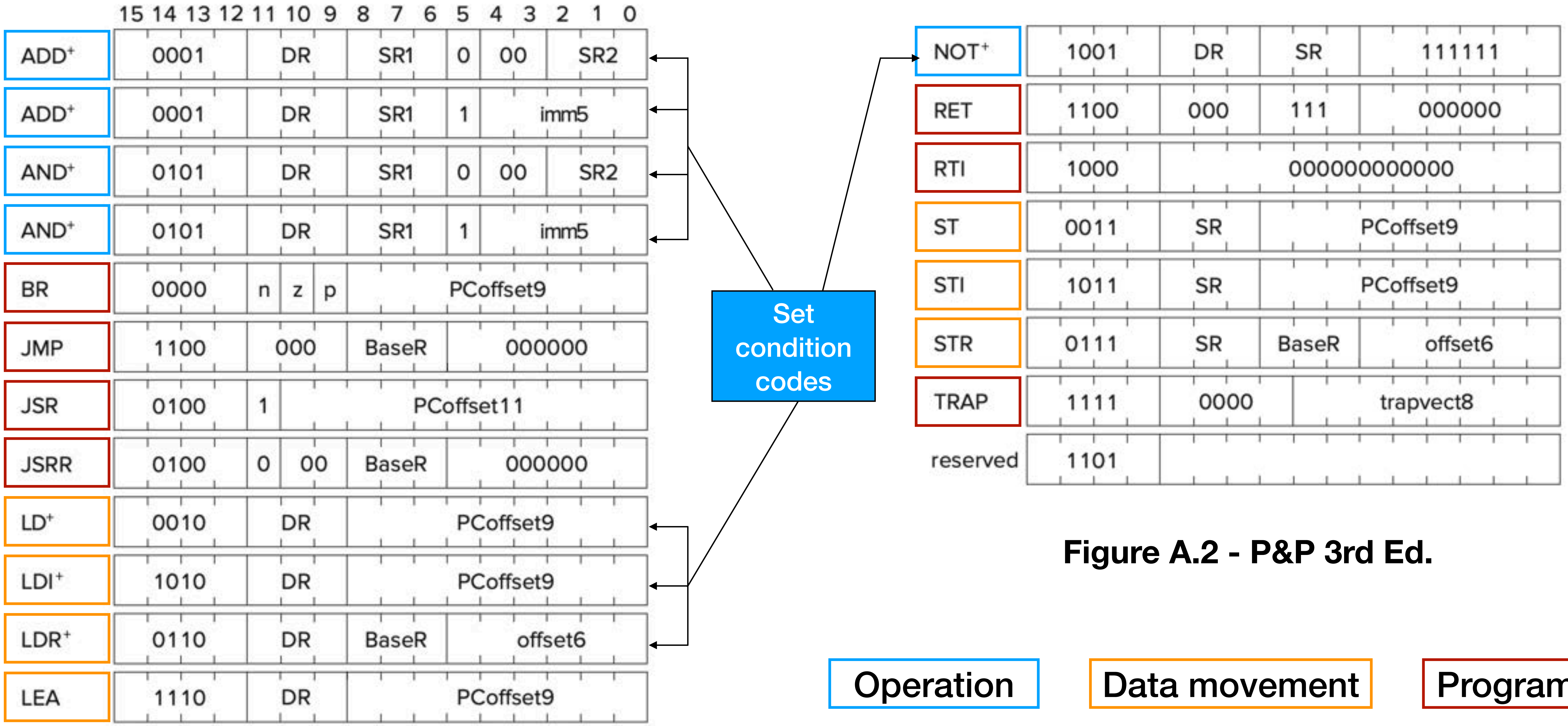
Figure 4.1 - P&P 3rd Ed.

LC3 Review

- Eight GPRs - denoted `R0, R1, …, R7`
- Data type: 16-bit 2’s complement integers
- Addressing: Locations `x0000 - xFFFF` contain 16 bits each
- Addressing modes:
  - Immediate, register, PC-relative, base + offset, indirect

LC3 - Review

Instruction set



15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

ADD$^+$	0001	DR	SR1	0	00	SR2
ADD$^+$	0001	DR	SR1	1	imm5	
AND$^+$	0101	DR	SR1	0	00	SR2
AND$^+$	0101	DR	SR1	1	imm5	
BR	0000	n	z	p	PCoffset9	
JMP	1100	000	BaseR	000000		
JSR	0100	1	PCoffset11			
JSRR	0100	0	00	BaseR	000000	
LD$^+$	0010	DR	PCoffset9			
LDI$^+$	1010	DR	PCoffset9			
LDR$^+$	0110	DR	BaseR	offset6		
LEA	1110	DR	PCoffset9			

NOT$^+$	1001	DR	SR	111111
RET	1100	000	111	000000
RTI	1000	000000000000		
ST	0011	SR	PCoffset9	
STI	1011	SR	PCoffset9	
STR	0111	SR	BaseR	offset6
TRAP	1111	0000	trapvect8	
reserved	1101			

Set condition codes

Figure A.2 - P&P 3rd Ed.

Operation

Data movement

Program flow

LC3 - Review

Addressing modes

- **PC relative**, the address is calculated by adding an offset to the incremented program counter, PC.
- **Register relative**, address is read from a register.
- **Indirect**, address is read from a memory location who”s address is calculated by adding an offset to the incremented program counter.
- **Load effective address** (LEA), address is calculated by adding an offset to the incremented program counter. The address itself (not its value) is stored in a register.

LC3 - Review

Addressing modes

Sign-extend (SX), by replicating the most significant bit as many times as necessary to extend to the word size of 16 bits.

Opcode	*Name*	*Assembly*	*Operation*
`LD`	Load	`LD DR, label`	`dr = mem[pc + SX(offset9)]`
`LDR`	Load Register	`LDR DR, BaseR, offset6`	`dr = mem[baseR + SX(offset6)]`
`LDI`	Load Indirect	`LDI DR, label`	`dr = mem[mem[pc + SX(offset9)]]`
`LEA`	Load Eff. Addr.	`LEA DR, target`	`dr = pc + SX(offset9)`
`ST`	Store	`ST SR, label`	`mem[pc + SX(offset9)] = sr`
`STR`	Store Register	`STR SR, BaseR, offset6`	`mem[baseR + SX(offset6)] = sr`
`STI`	Store Indirect	`STI SR, label`	`mem[mem[pc + SX(offset9)]] = sr`

Exercise

```
.ORIG x3000
 LD  R1, LABEL
 LDI R2, LABEL
 LDR R3, R2, #1
 LEA R4, LABEL
 LABEL .FILL x4001
.END
```

What are the values of `R1,R2,R3` & `R4` at each step?

Assume

```
; x4001 x6001
; ......
; x6001 x7001
; x6002 x7002
```

Answers

Ans	
R1	x4001
R2	x6001
R3	x7002
R4	x3004

Exercise

- Write a program to perform the multiplication 5 x 4.
  - Need a way to store 5 and 4 as arguments
  - There is no multiplication operation
    - So have to repeat addition

```
.ORIG x3000
; R0 - output, init to 0
; R1 - multipicand 1, init to 5
; R2 - loop counter, init to multiplicand 2

AND R0, R0, #0
AND R1, R1, #0
AND R2, R2, #0

ADD R1, R1, #5
ADD R2, R2, #4

LOOP    BRz DONE
        ADD R0, R0, R1
        ADD R2, R2, #-1
        BR LOOP

DONE HALT
.END
```

LC3 - Review

Pseudo-ops

- Looks like instruction but the “opcode” starts with a dot.
- Assembler instructions/directives that make our lives easier.

Opcode	*Operand*	*Meaning*
.ORIG	address	Starting address of program
.END		End of program
.BLKW	n	Allocate n words of storage
.STRINGZ	n-character string	Allocate n+1 locations, initialize with characters and null terminator

LC3 - Review

Microarchitecture

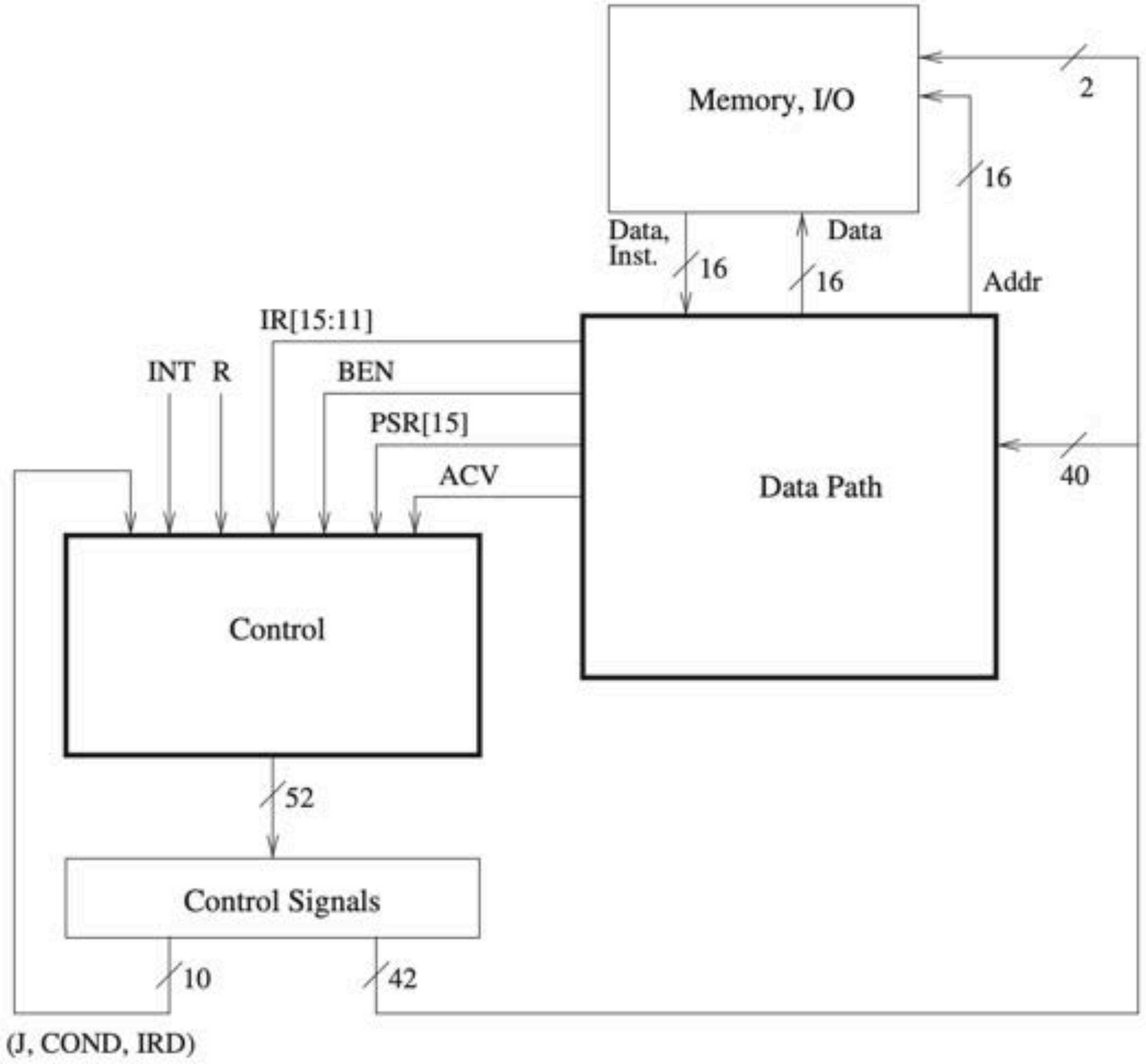


Figure C.1 - P&P 3rd Ed.

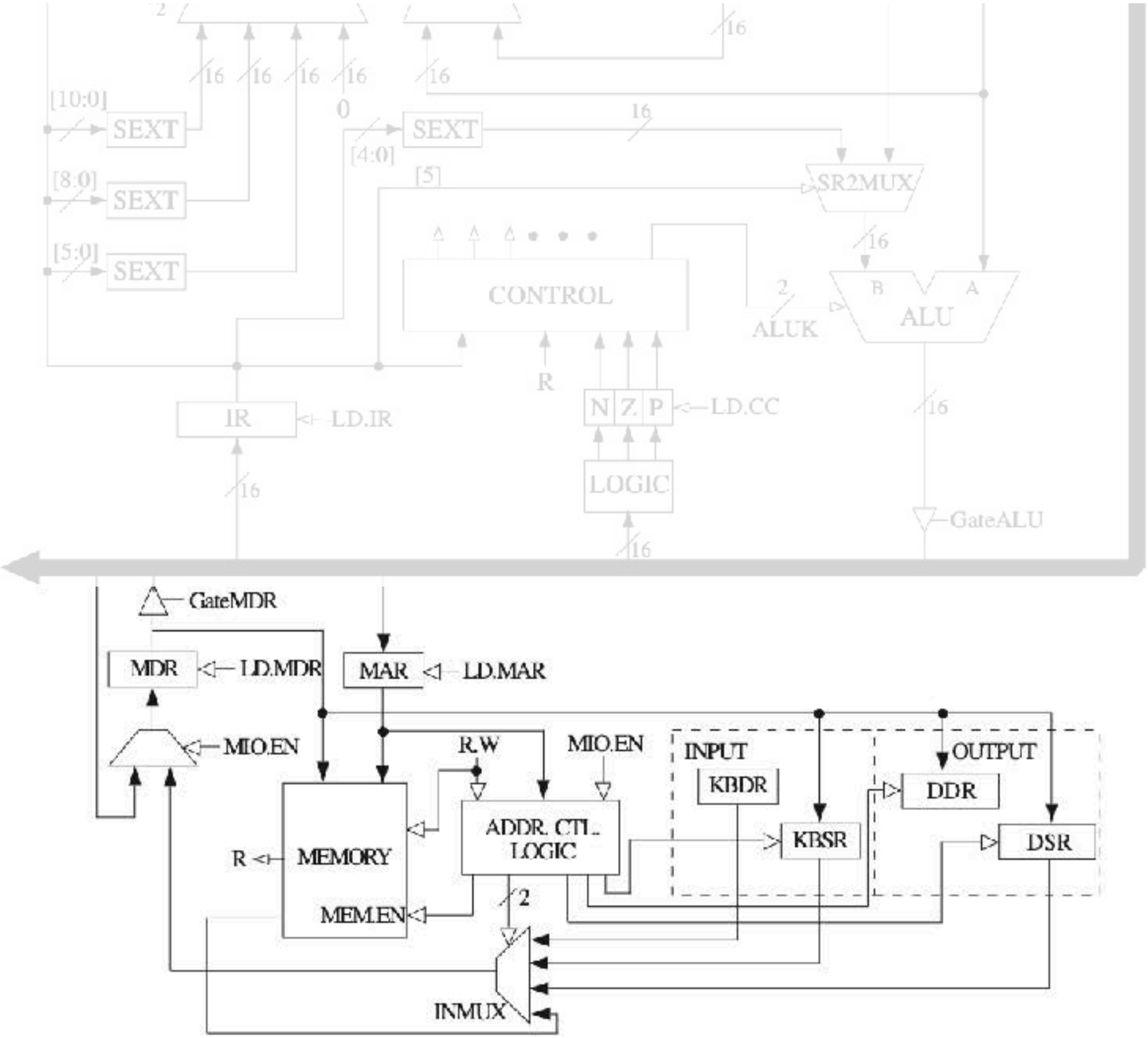


Figure C.3 - P&P 3rd Ed.

Textbook v2 vs. v3

- What is different in v3 compared to v2?
  - LEA no longer sets condition codes
  - TRAP instructions do not store linkage in R7

  This probably doesn't mean much to you right now

- What does that mean for you?
  - Do MPS on EWS machines
  - Practice for the quiz on the online simulator: https://courses.grainger.illinois.edu/ece220/sp2020/lc3web/index.html

Memory mapped I/O

- How do we communicate with the computer?

- Memory-mapped I/O: Hardware devices (i.e. their registers) are treated the same as the computer's main memory and addressable the same way

    - Memory of *peripherals* **is** *physically separate* from main memory

- Alternative: Port mapped I/O (requires having more specialized instructions)

- In LC3: `KBDR,KBSR,DSR,DDR` are used for [K]eyboard and [D]isplay respectively.

LC3 - Input/Output (IO)

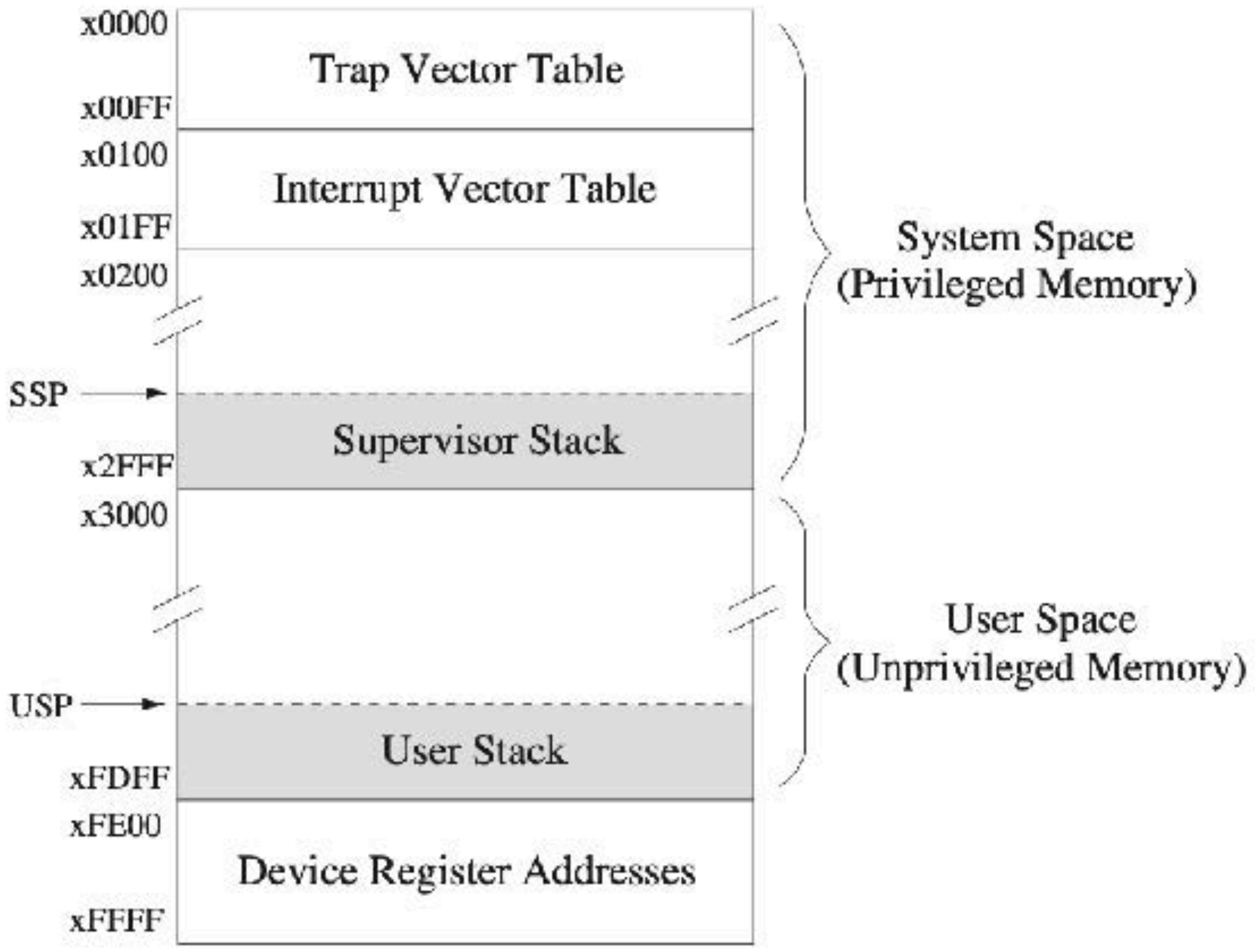
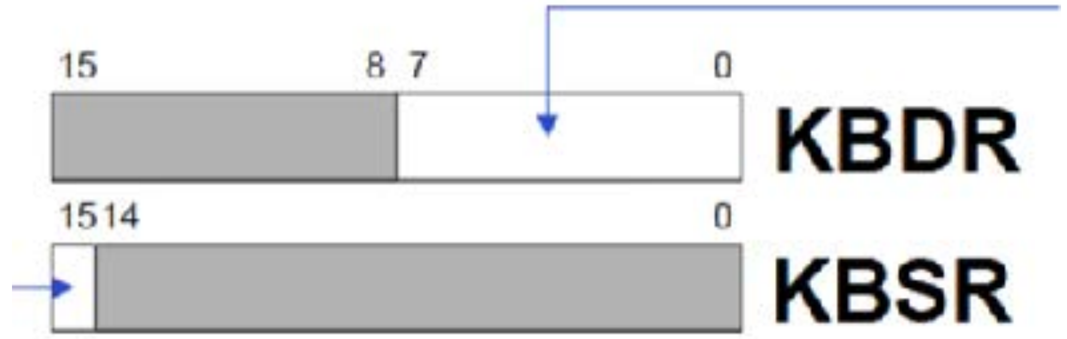
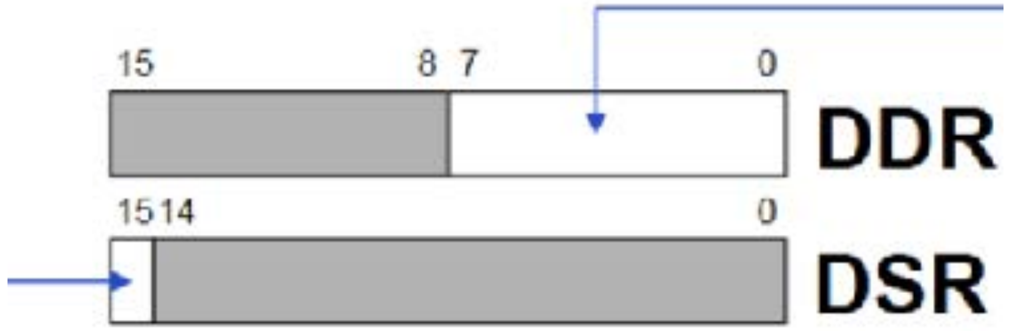


Figure A.1 - P&P 3rd Ed.

Address	I/O Register Name	I/O Register Function
xFE00	Keyboard status register (KBSR)	The ready bit (bit[15]) indicates if the keyboard has received a new character
xFE02	Keyboard data register (KBDR)	Bits [7:0] contain the last character typed on the keyboard
xFE04	Display status register (DSR)	The ready bit (bit[15]) indicates if the display device is ready to receive another character to print on the screen
xFE06	Display data register (DDR)	A character written in bits [7:0] will be displayed displayed on the screen

LC3 - Input/Output (IO)





Address	I/O Register Name	I/O Register Function
xFE00	Keyboard status register (KBSR)	The ready bit (bit[15]) indicates if the keyboard has received a new character
xFE02	Keyboard data register (KBDR)	Bits [7:0] contain the last character typed on the keyboard
xFE04	Display status register (DSR)	The ready bit (bit[15]) indicates if the display device is ready to receive another character to print on the screen
xFE06	Display data register (DDR)	A character written in bits [7:0] will be displayed displayed on the screen

LC3 - Input from keyboard

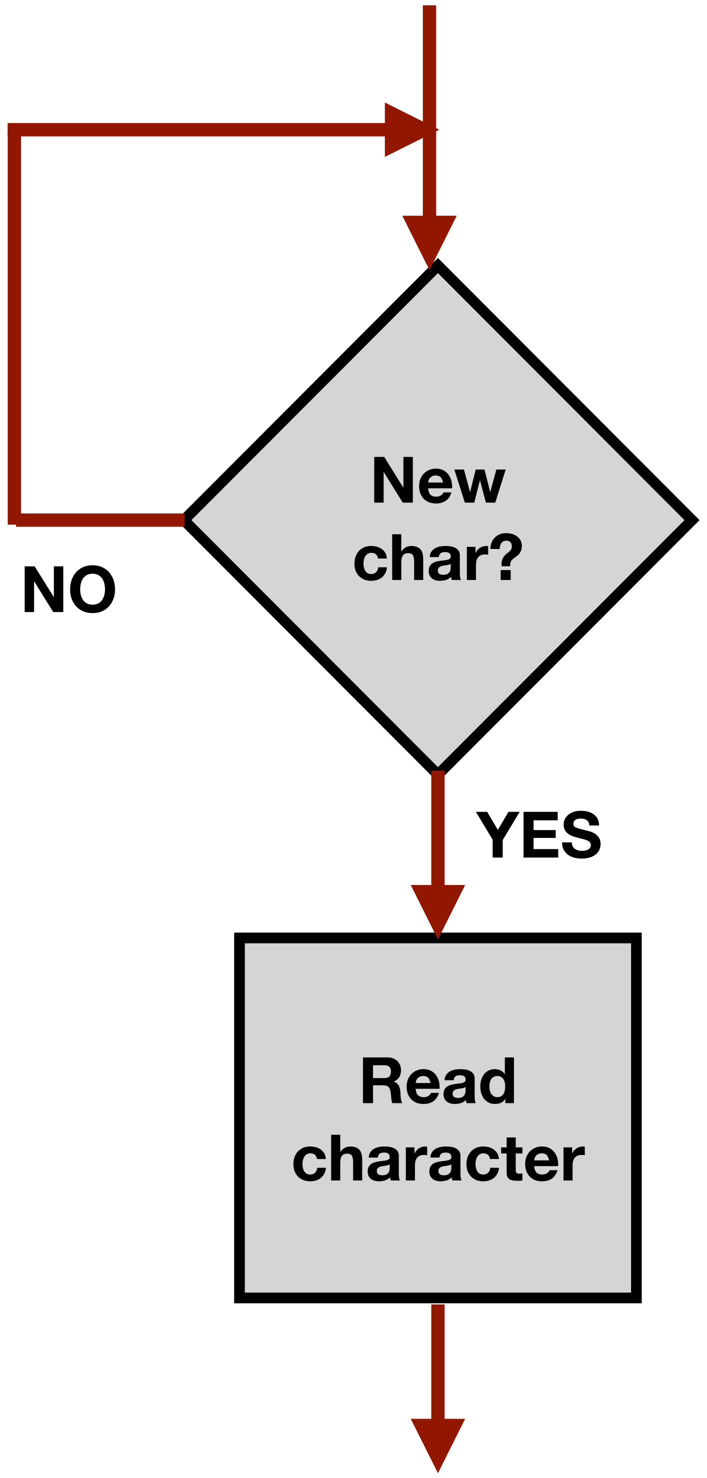
Basic routine

Handshaking is performed using `KBSR` & `KBDR`

- When user presses a key
  - Its ASCII code is placed in `KBDR[0:7]`
  - `KBSR[15]` is set to 1 *(ready bit)*
  - Keyboard is disabled, i.e., any further keypress is ignored
- When `KBDR` is read by CPU
  - `KBSR[15]` is set to 0
  - Keyboard is enabled

LC3 - Input from keyboard

Basic routine



```
.ORIG x3000

;Create a loop to
check KBSR


;If ready bit unset
loop again


;If ready bit set,
read KBDR into R0


KBSR .FILL xFE00

KBDR .FILL xFE02
```

LC3 - Display to console

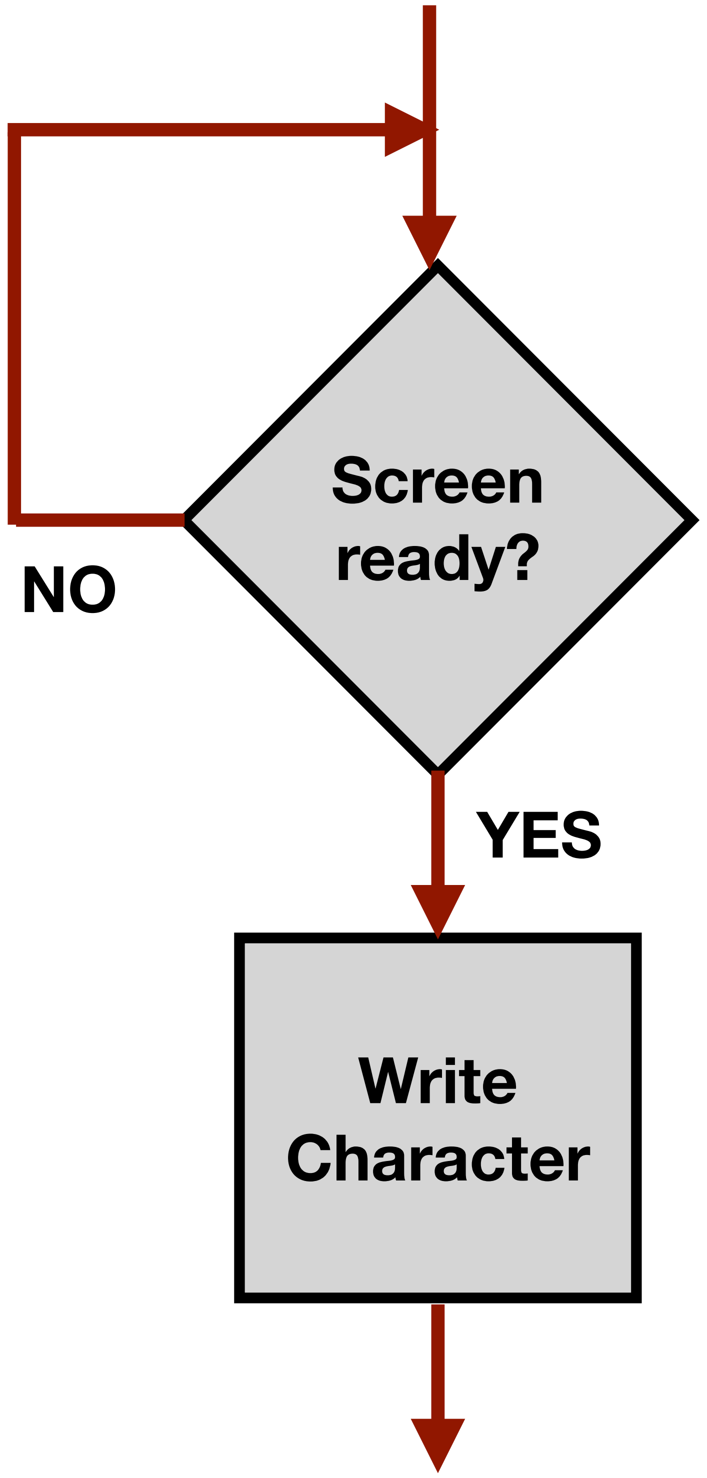
Basic routine

Handshaking is performed using `DSR` & `DDR`

- When display is ready to present a character
  - `DSR[15]` is set to 1 *(ready bit)*
- When a new character is written to `DDR`
  - `DSR[15]` is set to 0
  - Any other chars written to `DDR` are ignored
  - `DDR[7:0]` is displayed

LC3 - Display to console

Basic routine



```
.ORIG x3000

;Create a loop to
check DSR


;If ready bit unset
loop again


;If ready bit set,
write R0 into DDR


DSR .FILL xFE04

DDR .FILL xFE06
```

Exercise

- Write a program to display “ECE 220 is fun!” to the console. You can use the pseudo-op `.STRINGZ` to store string to memory. Do not use `TRAP` codes (*if you know what they are*).

```
.ORIG x3000
    LEA R2, MY_STRING
CHRLOOP  LDR R0, R2, #0
    BRz ALLDONE
    DPOLL
        LDI R1, DSR
    BRzp DPOLL
    STI R0, DDR
    ADD R2, R2, #1
BRnzp CHRLOOP

ALLDONE
    HALT

DSR  .FILL xFE04
DDR  .FILL xFE06

MY_STRING .STRINGZ  "ECE 220 is fun!"

.END
```

Issues?

- Limited amount of GPRs - polling display & keyboard uses up two of them
- Code often repeated - inefficient to keep inserting same code over & over again
- Human error - keeping track of registers & having direct access to hardware registers is recipe for unforced errors & bugs

Solution?

- Subroutines & repeated code
    - Also called *functions*
- TRAP routines
- More next time