ECE 220: Computer Systems & Programming

Lecture 1: Course Overview & LC-3 Review

Thomas Moon

January 16, 2024

About me

- Teaching at UIUC since 2019
- Instructed ECE 220 since 2019
- Also teach Signal Processing courses
  - ECE 310 (DSP)
  - ECE 420 (Mobile Application DSP Lab)
  - ECE 463 (IoT and Software-Defined Radio)
- And my research…

Can we break the Nyquist rate?

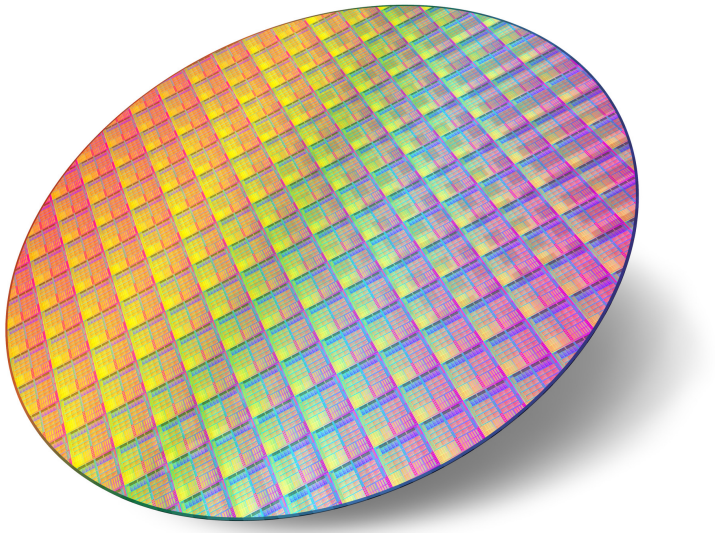
High-speed RF system

Radar security

Wireless sensing

Mobile DSP Algorithm

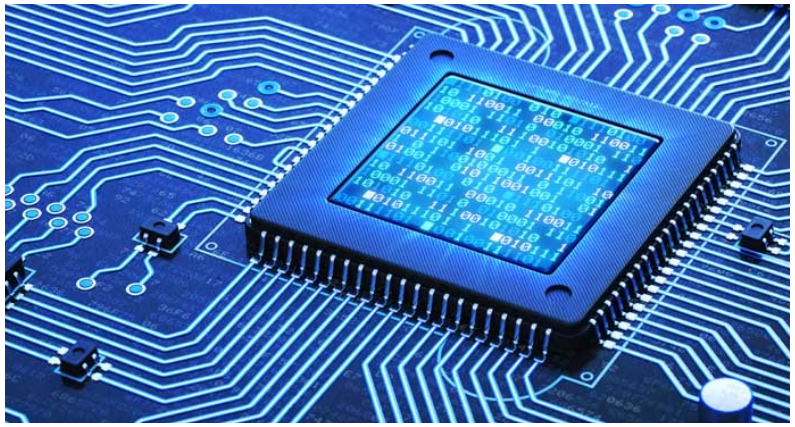
- I worked in IBM



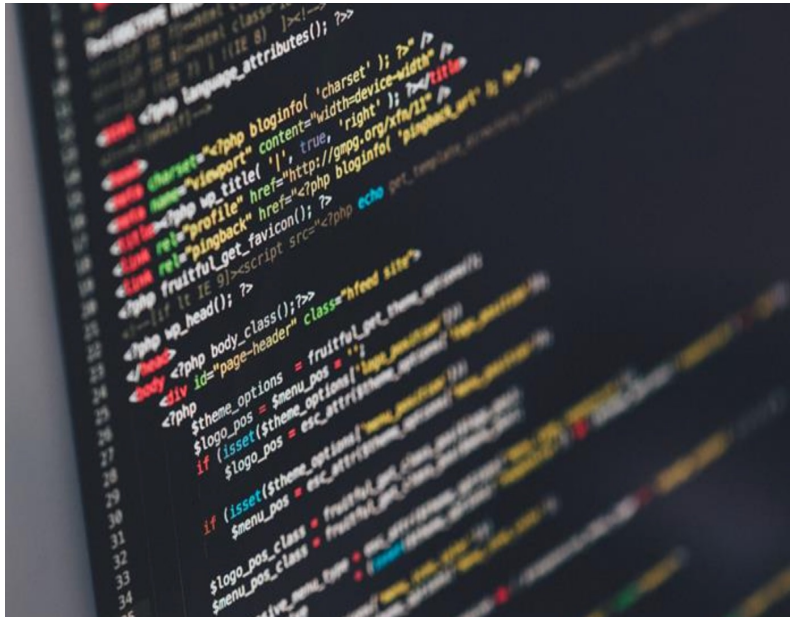
Automate RF system testing

C

ECE 220 is about



Computer Systems & Architecture



Programming C & C++

Course Outline – From 120 to 220

- C/C++ Programming
  - How do we write program in C or high-level language?
- I/O, TRAP, Stack, Interrupts
  - How does computer communicate with outside world?
- Assembly Language
  - How do we use instructions to implement algorithms?
- Instruction Set Architecture (LC-3)
  - What operations (instruction) do we implement?
- von Neumann Model
  - How do we construct basic computer model?
- Digital Logic
  - How do we build digital-circuits to process information?
- Bits and Data types
  - How do we represent information using electrical signal?

ECE220

ECE120

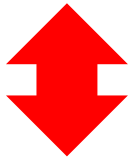
Course Logistics

https://courses.grainger.illinois.edu/ece220/sp2024/

At the end…

Part 1: LC-3

- Assembly language programming & process
- Memory-mapped I/O: input from keyboard, output to monitor
- TRAPs & Subroutines
- Stacks
- Interrupt-driven I/O: supervisor stack



Part 1.5: LC-3 to C

- Run-time stack, activation record
- Recursion in LC-3

Part 2: C

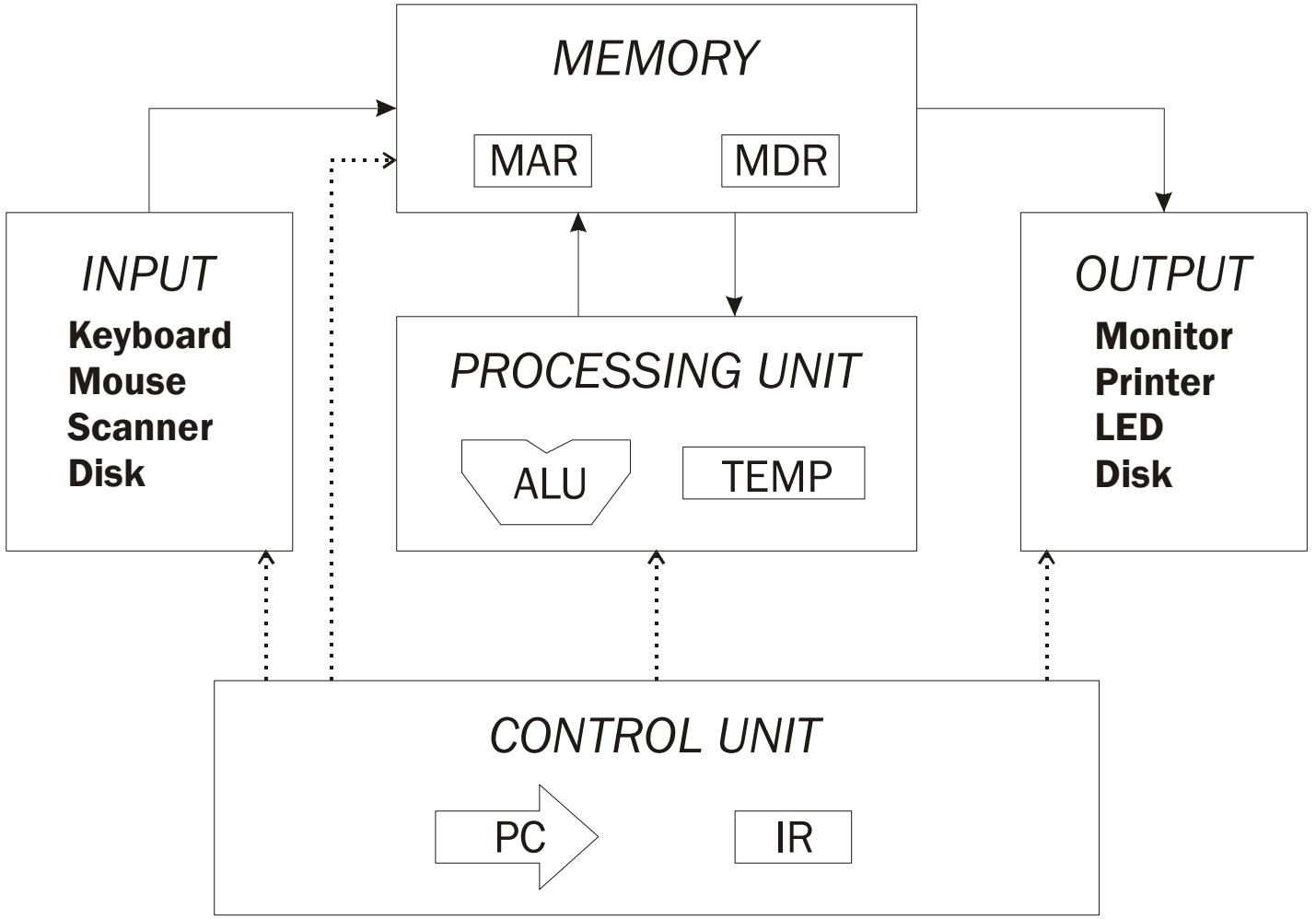
- Built-in data types, operators, scope
- Functions
- Pointers & arrays
- Recursion: searching, sorting, backtracking
- I/O: streams and buffers, read from / write to file
- User-defined data types: enum, union, struct
- Dynamic memory allocation
- Linked data structures: linked list (stack, queue) & trees

Part 3: C++

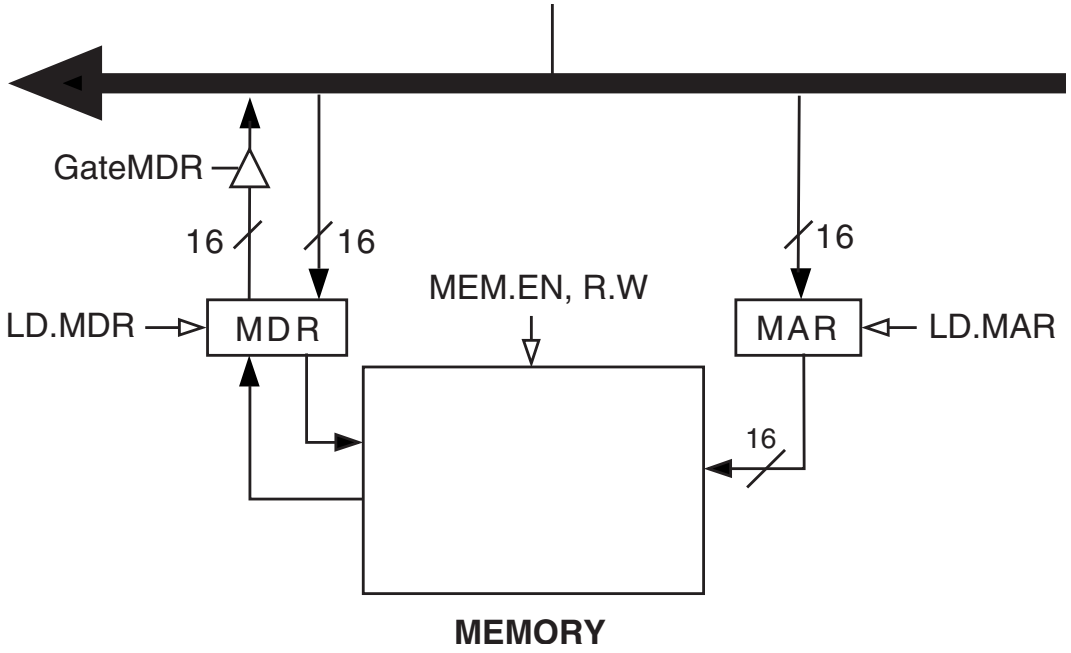
- Class (encapsulation, inheritance, abstraction, polymorphism)
- Virtual function, function/operator overload, template
- Pass by value / reference / address

LC-3 Review: von Neumann Architecture

1. Memory
2. Processing Unit
3. Control Unit
4. Input
5. Output

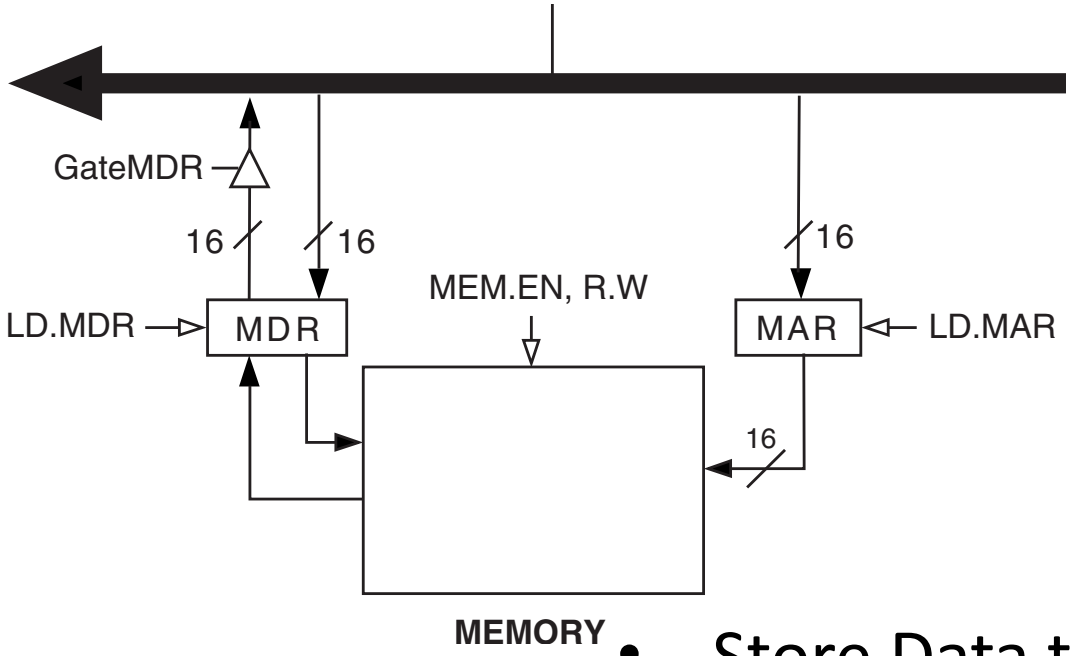


LC-3 Review: Memory



- Load and Store using
  - **MAR**: Memory Address Register (16-bit)
  - **MDR**: Memory Address Register (16 -bit)

LC-3 Review: Memory

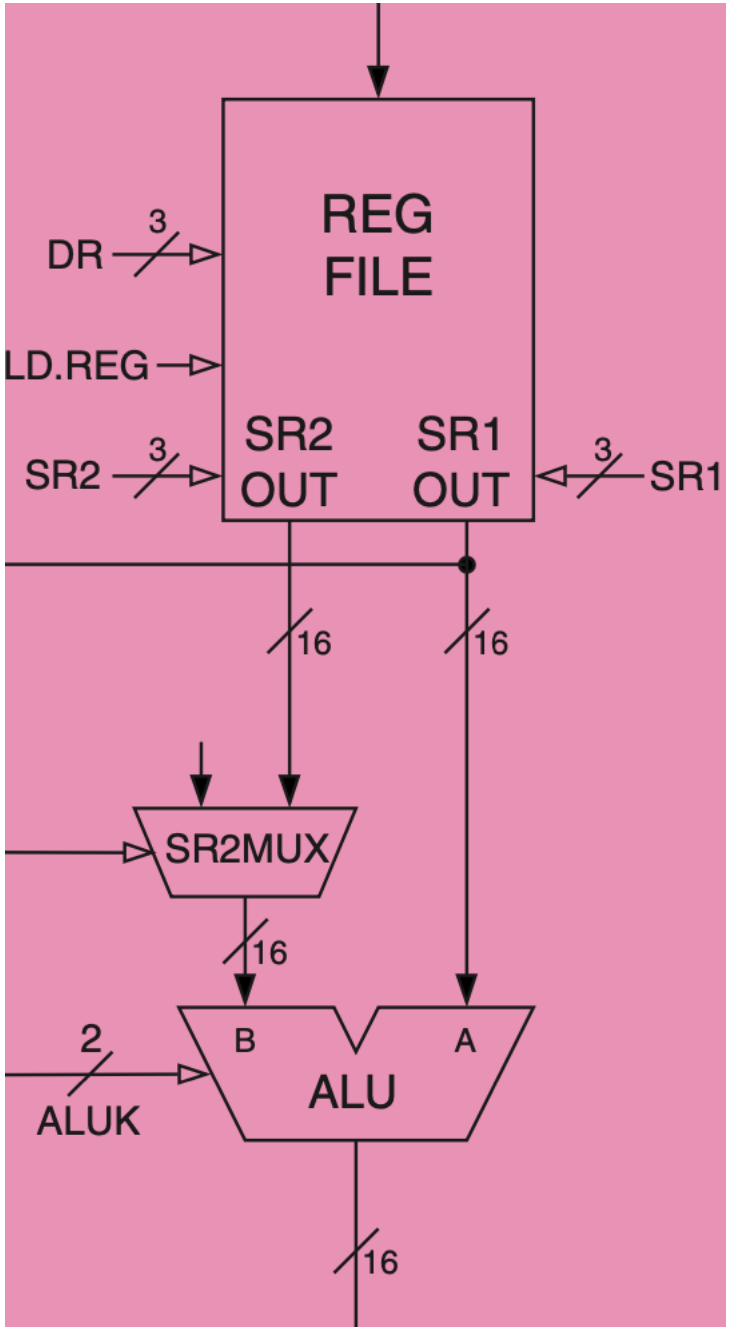


- Load Data from Memory Address X
  Step1:Place address X in MAR
  Step2:Send read signal to memory
  Step3:Data in X is placed in MDR

- Store Data to Memory Address Y
  Step1:Place address Y in MAR, place data in MDR
  Step2:Send write signal to memory
  Step3:Data is written to memory address Y

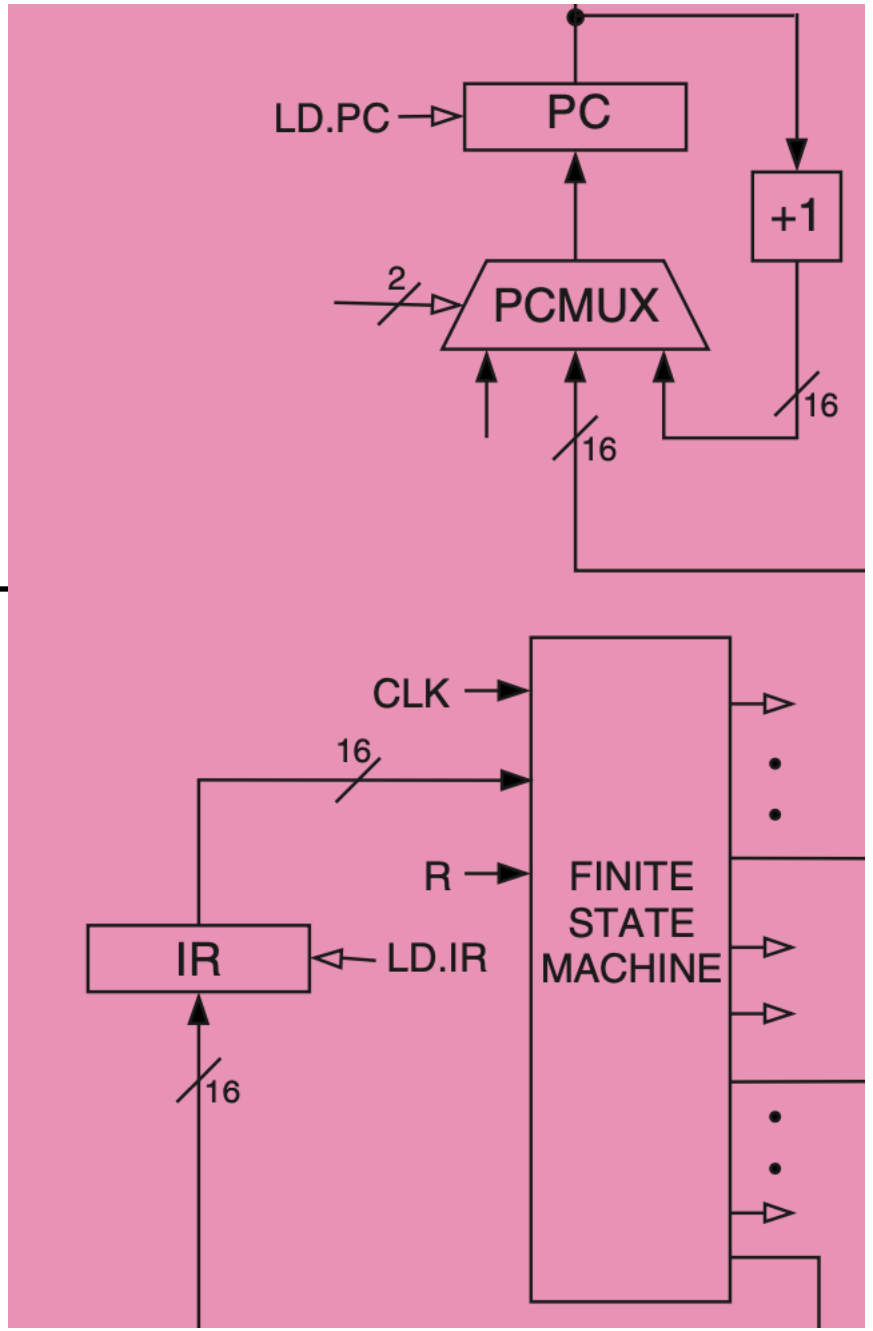
LC-3 Review: Processing Unit

- Processing Unit
  - The Arithmetic and Logic Unit (ALU) only has ADD, AND, NOT operations.
  - Temporary storage using general-purpose registers: R0 – R7

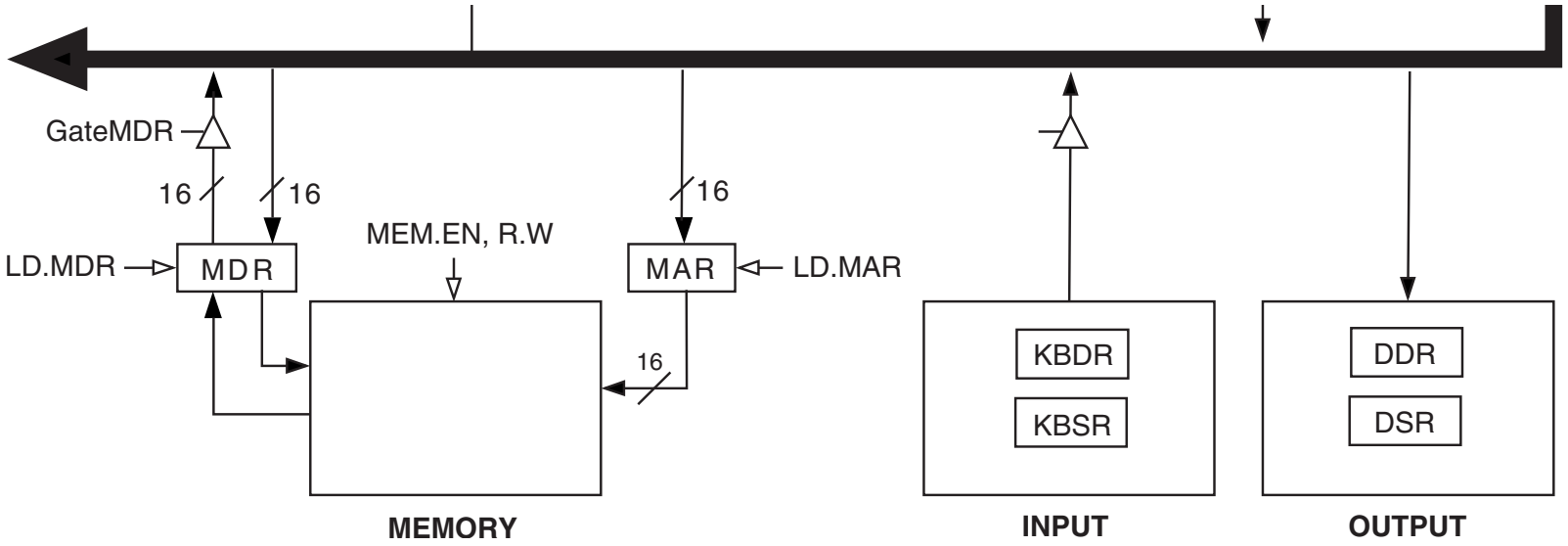


LC-3 Review: Control Unit

- Control Unit
  - IR: Instruction Register
    -Contains current instruction
  - PC: Program Counter
    -contains address of the next instruction



LC-3 Review: Input/Output



- Input – Keyboard (use 2 registers)
  1. KBDR (Keyboard Data Register)
  2. KBSR (Keyboard Status Register)
- Output – Monitor (use 2registers)
  1. DDR (Display Data Register)
  2. DSR (Display Status Register)

More details in lecture 2!

LC-3 Review: ISA (Instruction Set Architecture)

- ISA provides all information needed for someone who writes a program in machine language.

- ISA should tell you …
  - Memory organization
  - Register set
  - Instruction set
    - Data types
    - Addressing modes
    - Opcodes

LC-3 Review: ISA (Instruction Set Architecture)

Memory Organization

- Address space (# of distinct memory locations): 2^16
- Addressibility (# of bits stored in each memory location): 16 bit

Register set

- Eight 16-bit general-purpose registers: R0-R7
- Special-purpose register: IR , PC

MAR (2 bit): 4 (00, 01, 10, 11) < = 2^2

Q. A memory's addressibility is 64 bit. Which of the following is TRUE?

1. The address space is 2^{64} locations.
2. MAR is 64-bit.
3. MDR is 64-bit.
4. One address location stores 64-bit.

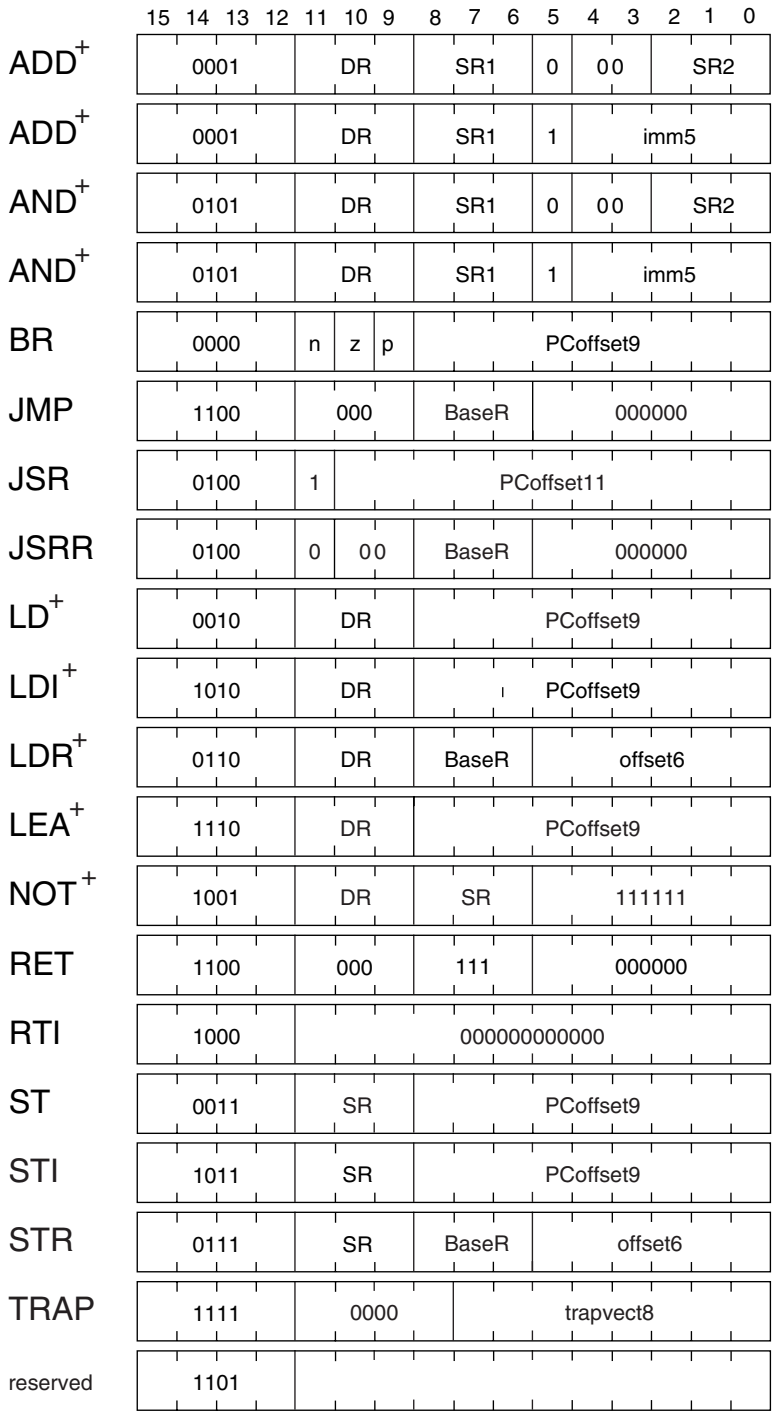
LC-3 Review: ISA (Instruction Set Architecture)

Instruction Set

- **Data Types**
  - 16-bit 2's complement integers

- **Addressing modes** (how the location of operand is specified)
  - Non-Memory addresses – Immediate (part of instruction), register
  - Memory address – PC-relative, base+offset, indirect

- **Opcodes** (15 opcodes, bits[15:12] used to specify the opcode)
  - Operate instructions: ADD, AND, NOT
  - Data movement instructions: LD, LDI, LDR, LEA, ST, STR, STI
  - Control instructions: BR, JSR/JSRR, JMP/RET, TRAP, RTI


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

LC-3 Review: Pseudo-ops

- .ORIG *<address>*
  - Specifies where in the memory the program should be placed
- .END
  - Indicates the end of the program text
  - Has nothing to do with stopping the program
- .FILL *<n>*
  - Allocate one word initialized with *<n>*

```
.ORIG x3000
.FILL xABCD
.FILL xFEDC
.END
```

Memory

Address	Data
x3000	xABCD
x3001	xFEDC

LC-3 Review: Pseudo-ops

- .BLKW *<n>*
  - Allocates *<n>* words initialized with x0

```
.ORIG x3000
.FILL xABCD
.BLKW #2
.FILL xFEDC
.END
```

Address	Data
x3000	xABCD
x3001	x0
x3002	x0
x3003	xFEDC

- .STRINGZ *<"n-character string">*
  - Allocates *n+1* words initialized with characters and null terminator

```
.ORIG x3000
.STRINGZ "test"
.END
```

Address	Data
x3000	x74
x3001	x65
x3002	x73
x3003	x74
x3004	x0

LC-3 Review: LD family

```
        .ORIG       x3000
        LD          R2,LB
        LDI         R3,LB
        LDR         R4,R2,#1
        LEA         R5,LB
LB      .FILL       x00FF
        .END
```

R2 = x00FF , R2 <- mem[LB (x3004)]

R3 = x0495 , R3<- mem[mem[LB]]

R4 = x0498, R4 <- mem[R2 + 1]

R5 = x3004, R5<- address of LB

Address	Data
x00FF	x0495
X0100	x0498

MP1: Printing a histogram

- Goal
  - Count the occurrence of each letter (A to Z) in an ASCII string
  - Lower case and upper case are counted together
  - Count also non-alphabetic characters
  - Print the histogram in hexadecimal (your code)
  - Avoid using R7

Input string in memory

Address	contents
X4000	'E'
X4001	'c'
X4002	'e'
X4003	'2'
X4004	'2'
X4005	'0'
X4006	NUL

Output histogram in memory

Address	contents	comments
X3F00	x3	;non-alph
X3F01	x0	;'a'
X3F02	x0	;'b'
X3F03	x1	;'c'
X3F04	x0	;'d'
X3F05	x2	;'e'
X3F06	x0	;'f'

What your code does…

Output histogram in memory

```
Address     contents    comments
X3F00       x3          ;non-alph
X3F01       x0          ;'a'
X3F02       x0          ;'b'
X3F03       x1          ;'c'
X3F04       x0          ;'d'
X3F05       x2          ;'e'
X3F06       x0          ;'f'
```

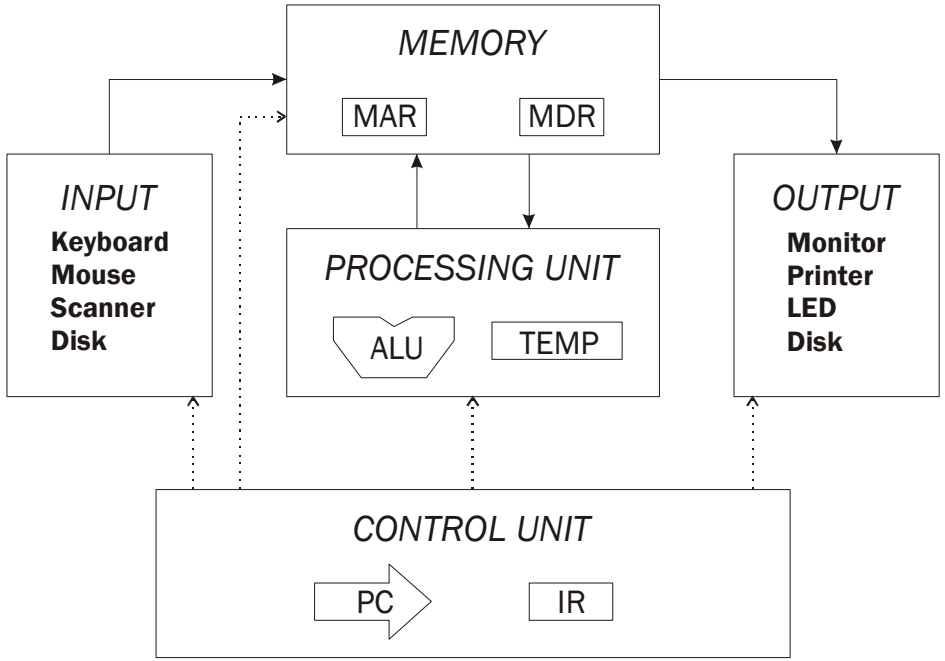
Final output in display

```
@ 0003
A 0000
B 0000
C 0001
D 0000
E 0002
F 0000
```

Takeaways

https://courses.grainger.illinois.edu/ece220/sp2024/

LC-3



KBDR
KBSR
DDR
DSR