EEE3096S: Embedded Systems II

LECTURE 30:

CPU ARCHITECTURE AND DATAPATH (PART 2)

Presented by:

STANLEY MBEWE

Electrical Engineering
University of Cape Town

OUTLINE OF LECTURE

- Computer Design Fundamentals
- Building the Instruction Set
- The Function Unit in a CPU
- The ALU in a CPU
- Datapath pass 2: understanding the connection of the parts CU -> FU -> ALU
- Deciding the instruction set for a given processor design

Linked reading, for going deeper into these topics that are more introduced in these lectures:
Chapter 2 of Textbook: D. Patterson and J. Hennessy - Computer Organization and Design - ARM Edition.pdf

Computer Design Fundamentals

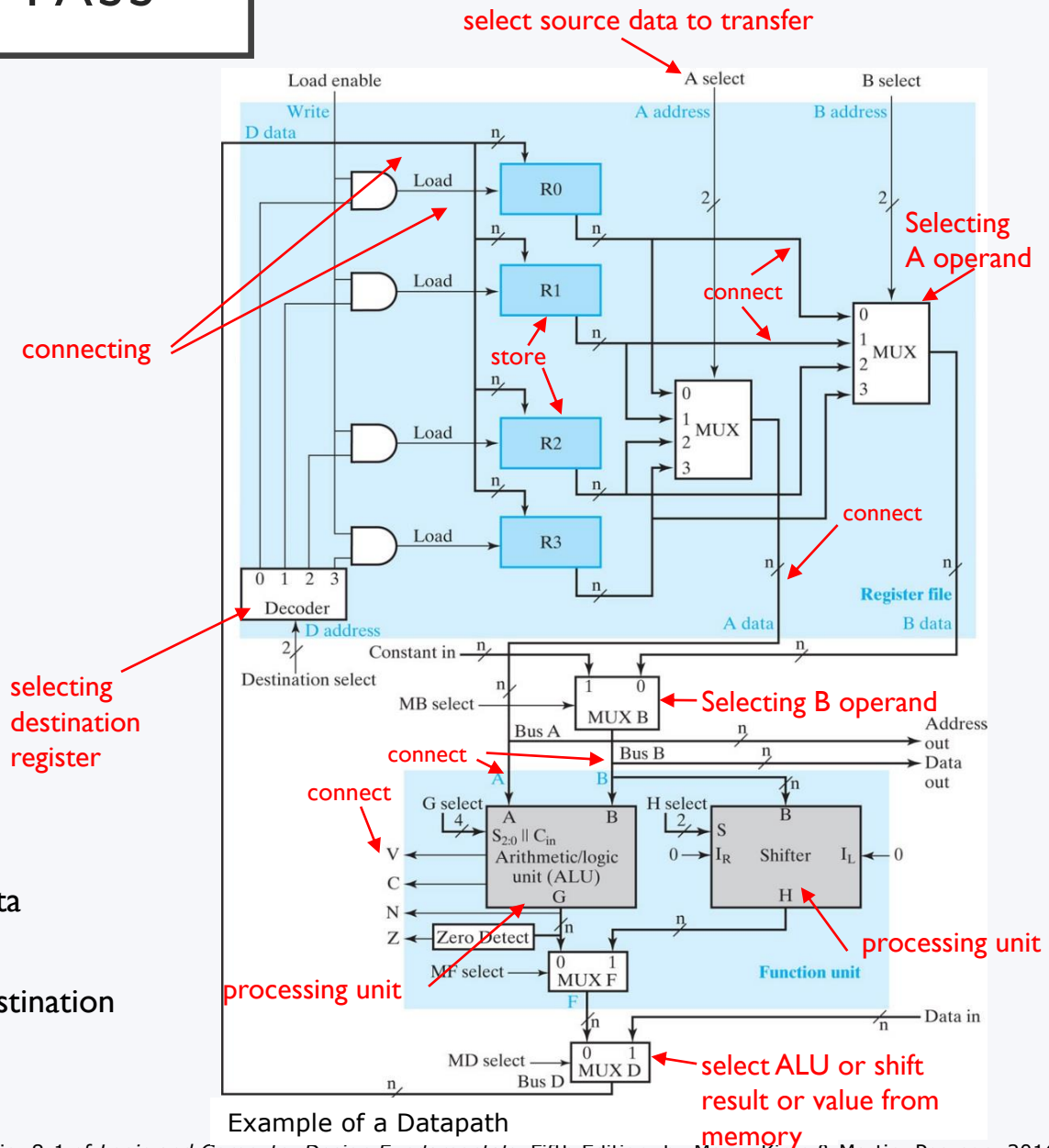
DATAPATH – 1ST PASS

Datapath design is largely about the circuitry to store data and to select and connect data to be transferred between processing units in a processor

Will briefly explain the parts…

1. 4x registers available in this arch
2. Select destination & source for data
3. Connections between the parts
4. Move result or shifted value to destination

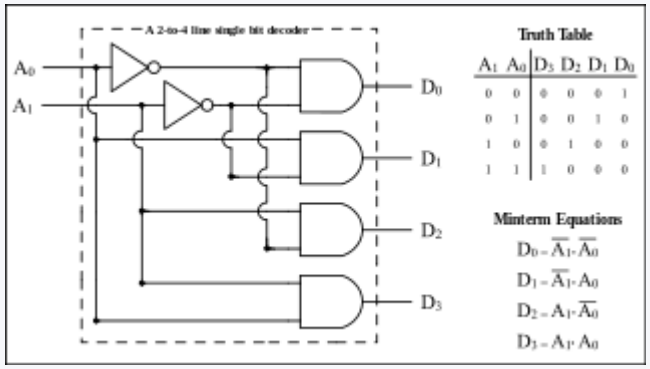
Onwards to….



Example of a Datapath

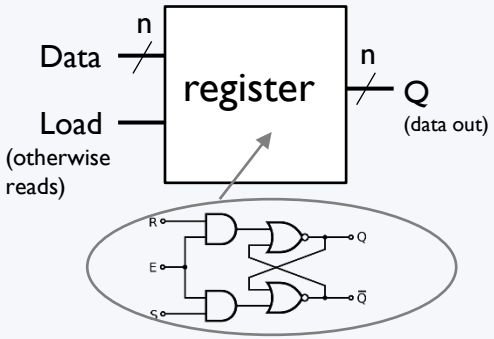
Based on CH8 Fig. 8-1 of *Logic and Computer Design Fundamentals*, Fifth Edition, by Mano, Kime & Martin. Pearson. 2016.

BRIEF REFRESHER – PROCESSOR PIECES



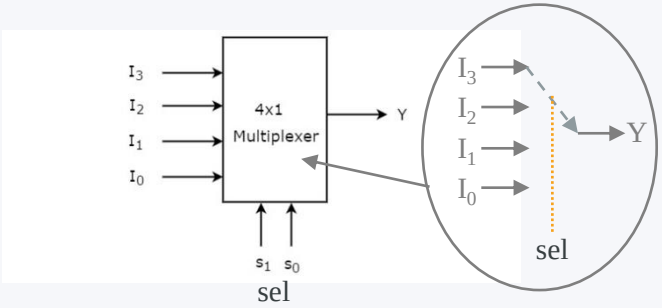
The Decoder

'Fans out bits'
i.e. convers m-bit input to activating one of 2^m output signals. This is used for decoding an instruction, to chip select one of multiple operations



The Register

Stores values. Basically a D-type flip flop that can be enabled(load)/disabled(read)



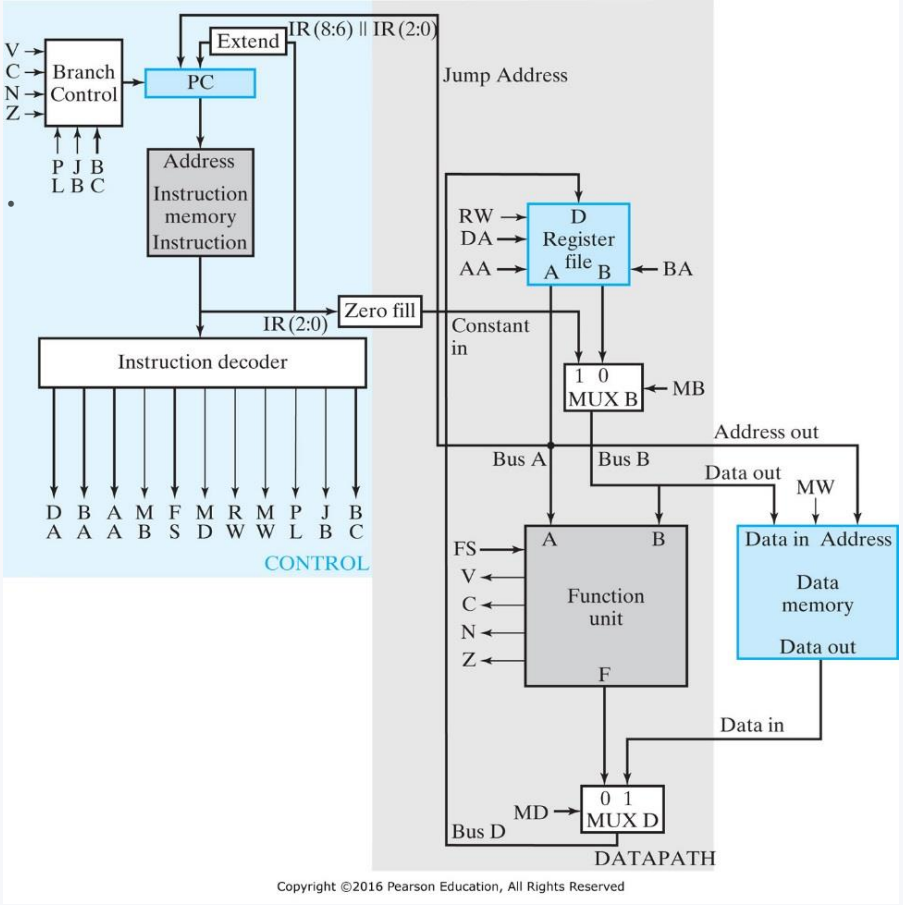
The Multiplexer

m-input sel line to select 1 of 2^m input lines (I) to connect through to the output line (Y)

INSTRUCTION SET ARCHITECTURE (ISA)

General Computer Architecture comprises:

- Instruction Set Architecture* …
- *Storage Resources:*
  - Instruction memory,
  - Data memory,
  - Register file,
  - Program counter
- *Datapath:*
  - Runs the instructions and activates processes

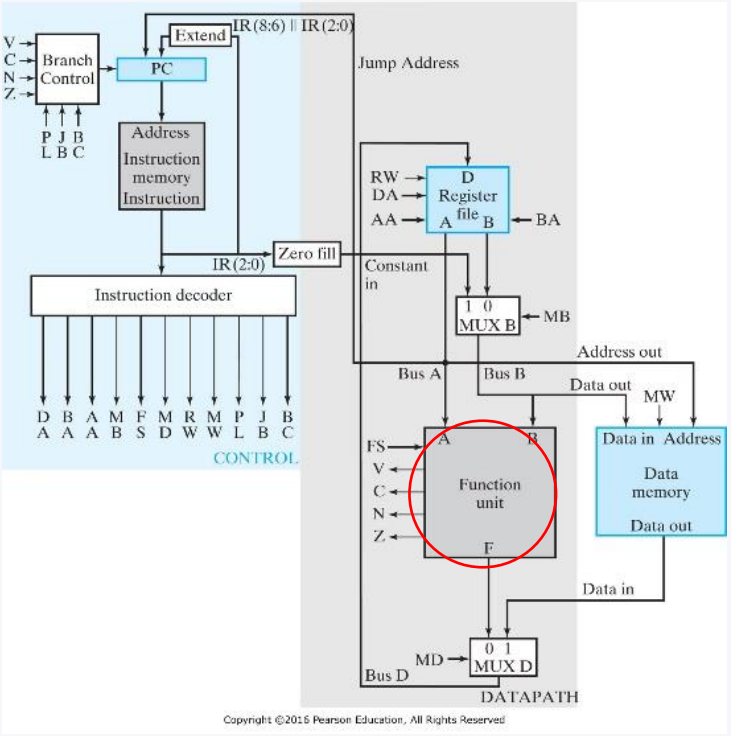


Based on CH8 Fig. 8-15 of *Logic and Computer Design Fundamentals*, Fifth Edition, by Mano, Kime & Martin. Pearson. 2016.

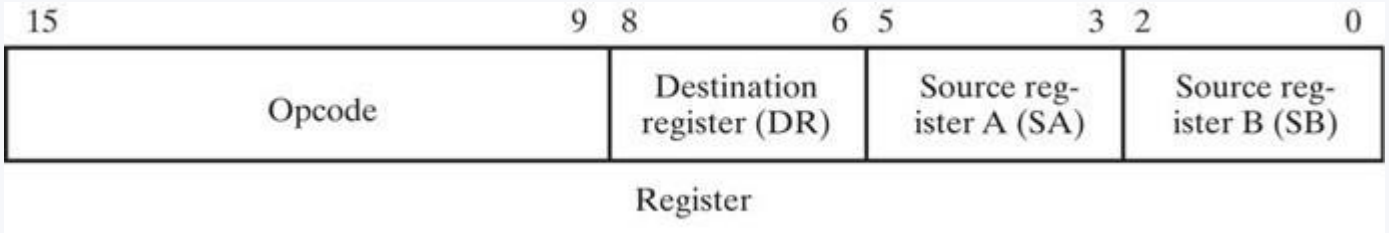
BUILDING THE INSTRUCTION SET

Instruction Set Architecture (ISA) =

- *Instruction format:* how the instructions are structured, how the bits of the instruction link to operations/data
- *Instruction specification:* describe each of the available instructions that the system can execute.



Block diagram of a single cycle computer



An example instruction format that is likely used for arithmetic/logic operations on a RISC processor

* we delve into detail later in the course

Based on CH8 Fig. 8-15 of *Logic and Computer Design Fundamentals*, by Mano, Kime & Martin. Pearson. 2016.

THE FUNCTION UNIT

Function Unit (FU)
(or 'execution unit')

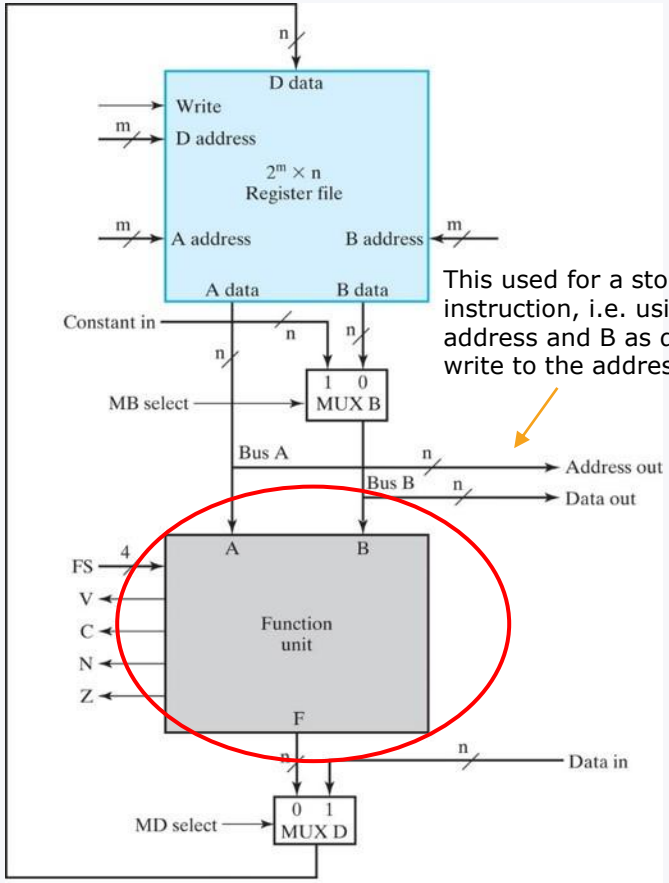
- Carries out the operations
- Could be structured in various ways.
- Typically…

FU = ALU + CU

and possibly

$FU_{main} = \Sigma (FU_{subunits})$

for a multi-processor system



This used for a store instruction, i.e. using A as address and B as data to write to the address

Block diagram of a single cycle computer

As show in this diagram, the FU has two input registers, A and B, that it operates on. There is a Function selector (FS) input used to chose the operation to perform. When this operation is performed, the result output is latched to F and updates status flags V,C,N,Z. The MD (memory data) select can bypass the FU if the instruction is loading from memory.

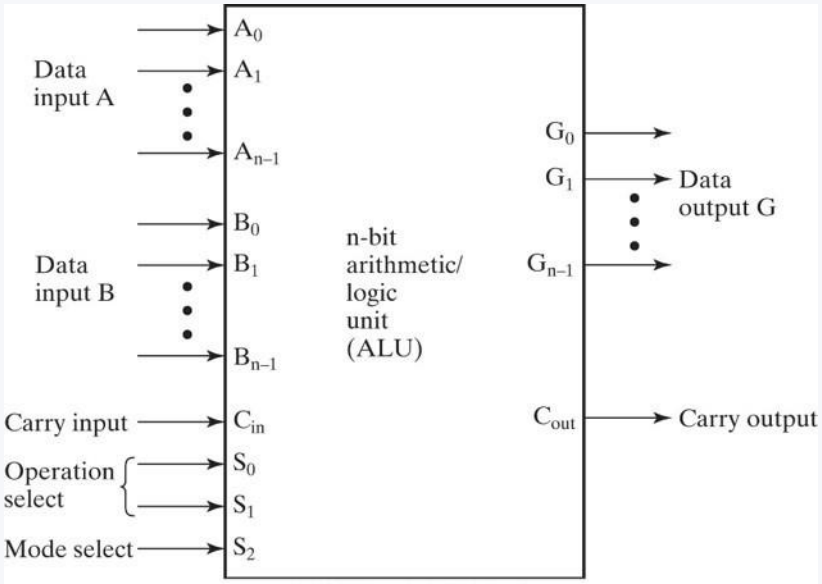
Based on CH8 Fig. 8-15 of *Logic and Computer Design Fundamentals*, by Mano, Kime & Martin. Pearson. 2016.

THE ALU

- The Arithmetic/Logic Unit (ALU)
  - Arithmetic operations: + - * /
  - Logic operations: and, or, xor, not

The ALU has

- Inputs to provide data to be processed as well as
- Input for type of operation to be performed, (e.g. if it needs to do an ADD or AND, etc.)
- Has outputs that provide the results of the operation (G output in figure).
- Output flags, e.g. carry (C) output flag that stores status information of the operation (e.g. carry flag set) or error states (divide by zero).



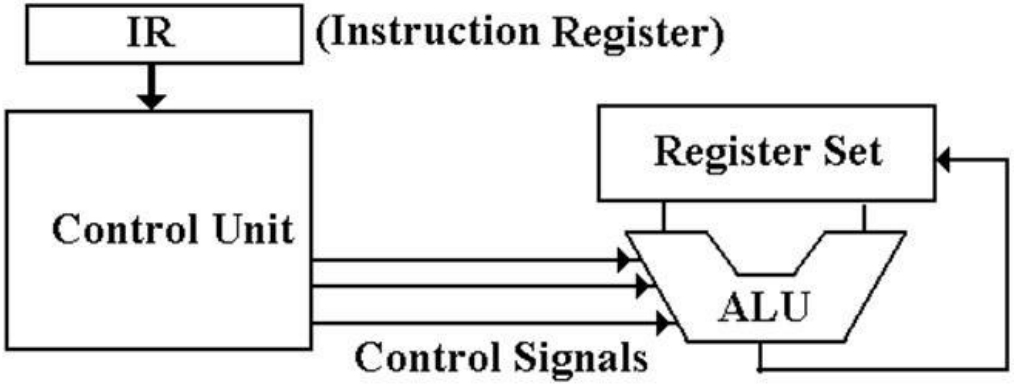
An *n*-Bit ALU

Based on CH8 Fig. 8-2 of *Logic and Computer Design Fundamentals*, by Mano, Kime & Martin. Pearson. 2016.

THE CONTROL UNIT

- Control Unit (CU)
  - The CU is the part of the CPU that directs the working of the processor.
  - The CU coordinates the computer's memory, ALU and its input and output devices, making these parts respond to program instructions.

Sometimes the control unit is not shown in the design of a computer (or cannot be shown as a distinct subsystem) as it may be distributed through the system, closely integrated within other components; this is particularly the case for highly complex pipelined architectures where it is difficult to have a separate component to arbitrate the system.



Control Unit: can be abstracted as the circuitry used to sequence and control the system (this simplification obviously hides many aspects such as memory and IO)

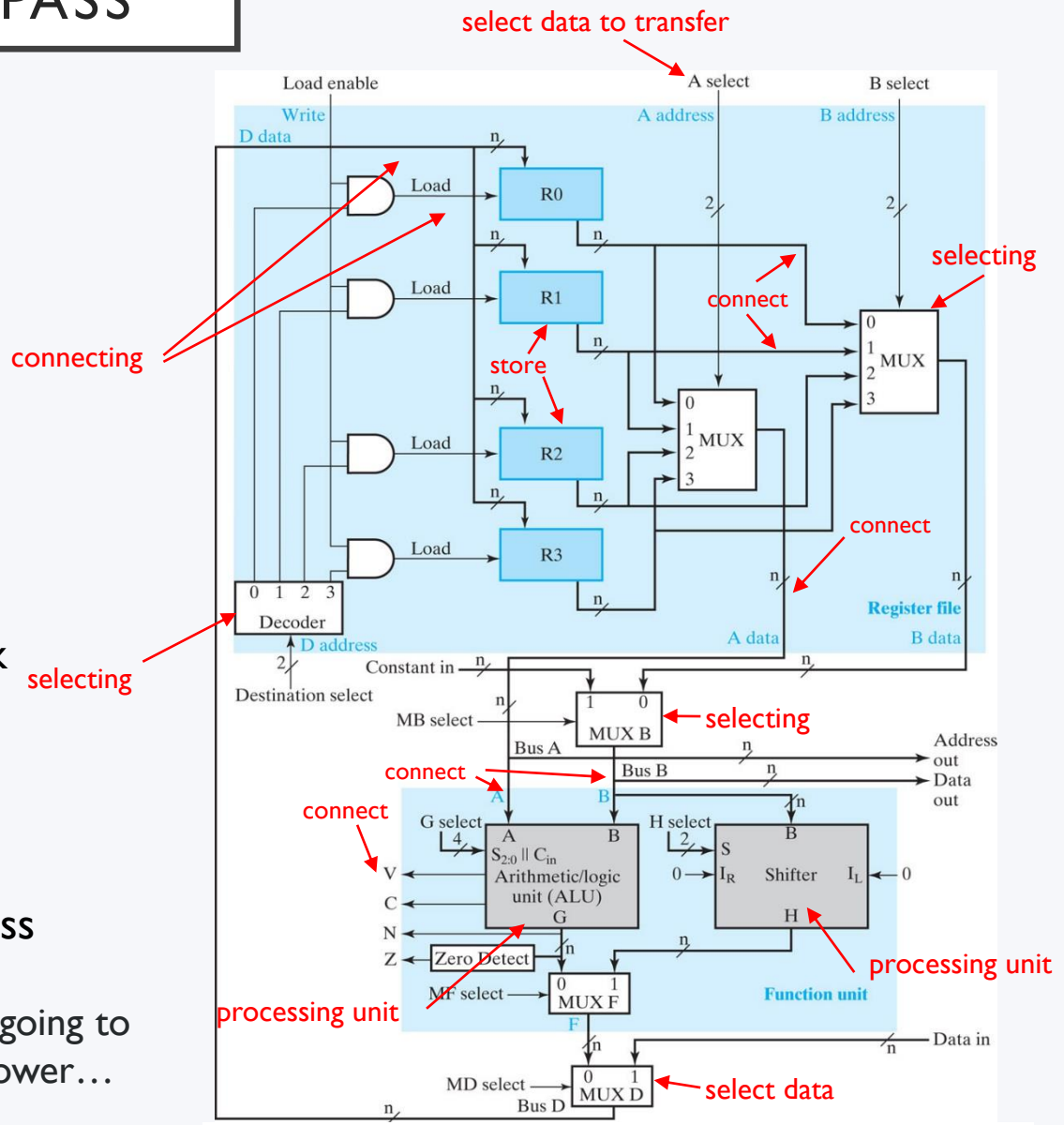
DATAPATH – 2ND PASS

We've looked at the datapath to connect the pieces, the register file, the ALU, the FU, the CU.

Now you hopefully have an inkling of how the pieces work together to

1) get data to the FU inputs,
2) grab results from FU,
3) move FU results/memory to register or memory address

Don't be too worried yet: we are going to run through the process a little slower…



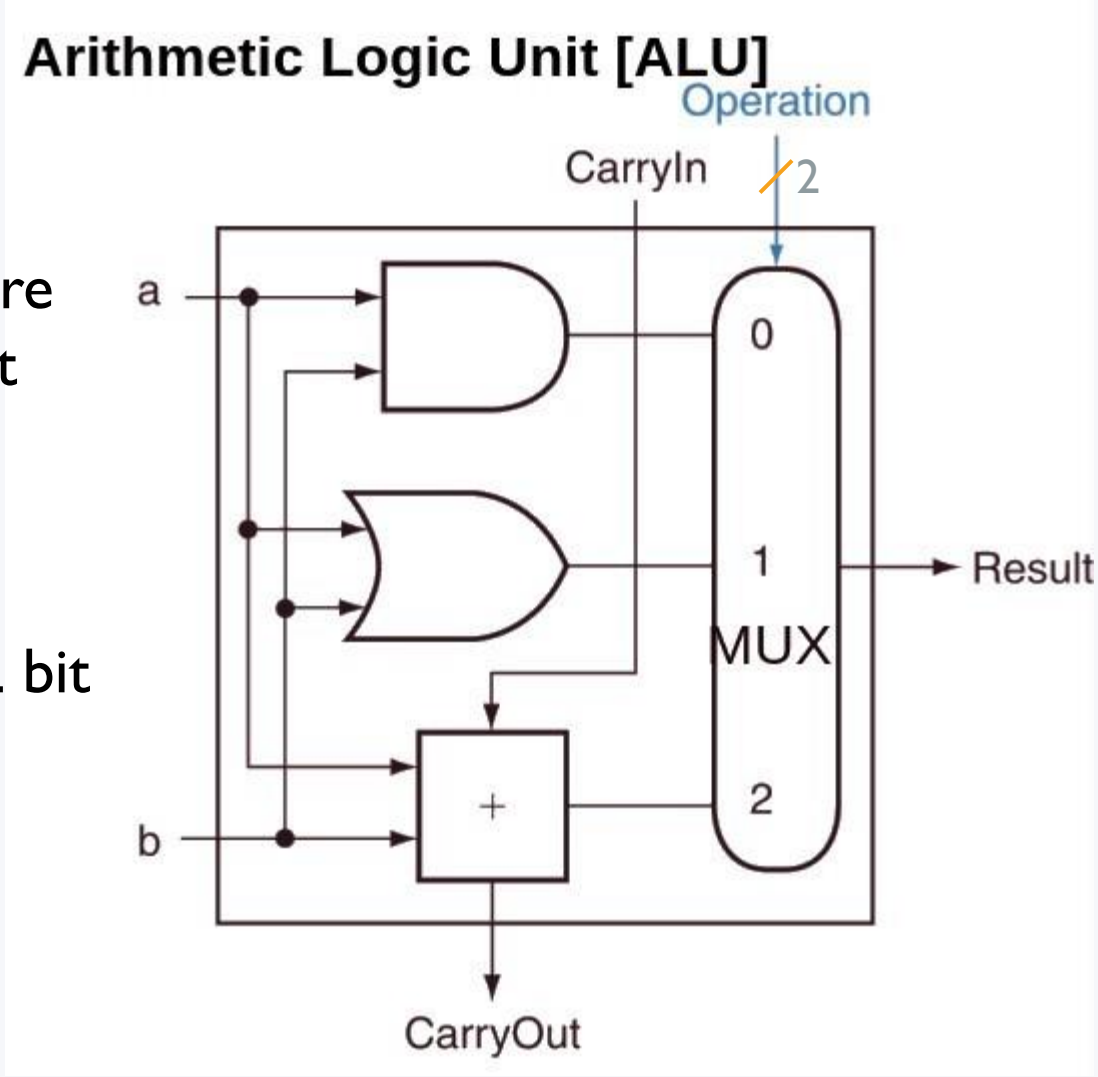
Example of a Datapath

Based on CH8 Fig. 8-1 of *Logic and Computer Design Fundamentals*, Fifth Edition, by Mano, Kime & Martin. Pearson. 2016.

Processor Pieces

A simple ALU Example

This is a simplified ALU example, where there are two input operands (a,b), a carryIn bit and a carryOut bit. The Operation line (a 2 bit bus here) selects which operation (AND, OR or +) is applied.

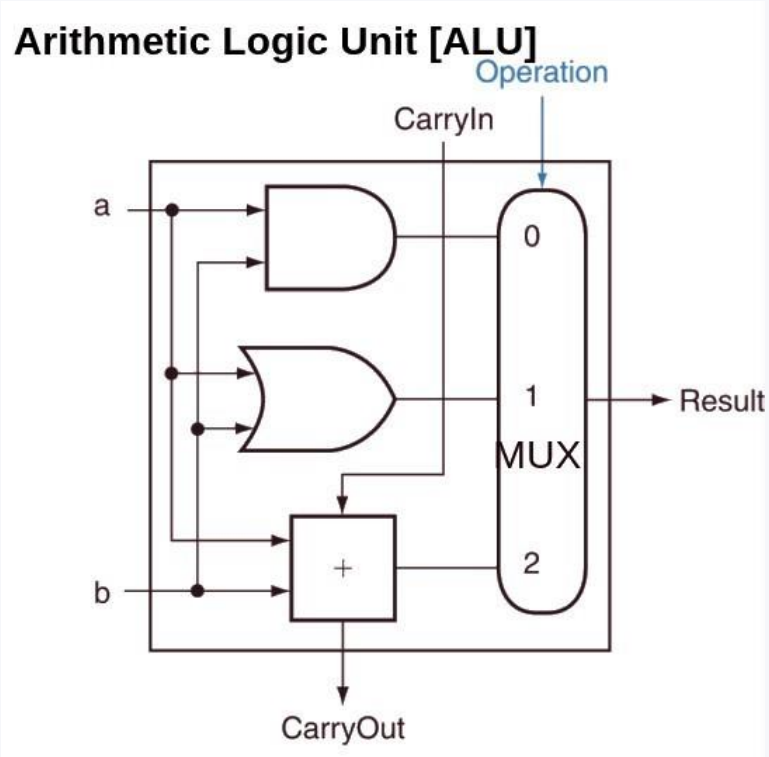


DECIDING PROCESSOR INSTRUCTIONS

- You can either decide the instructions and their names first, or do that after you have the processor designed.
- Let us apply the second approach (not yet having named instructions) to illustrate the process or deciding instructions …

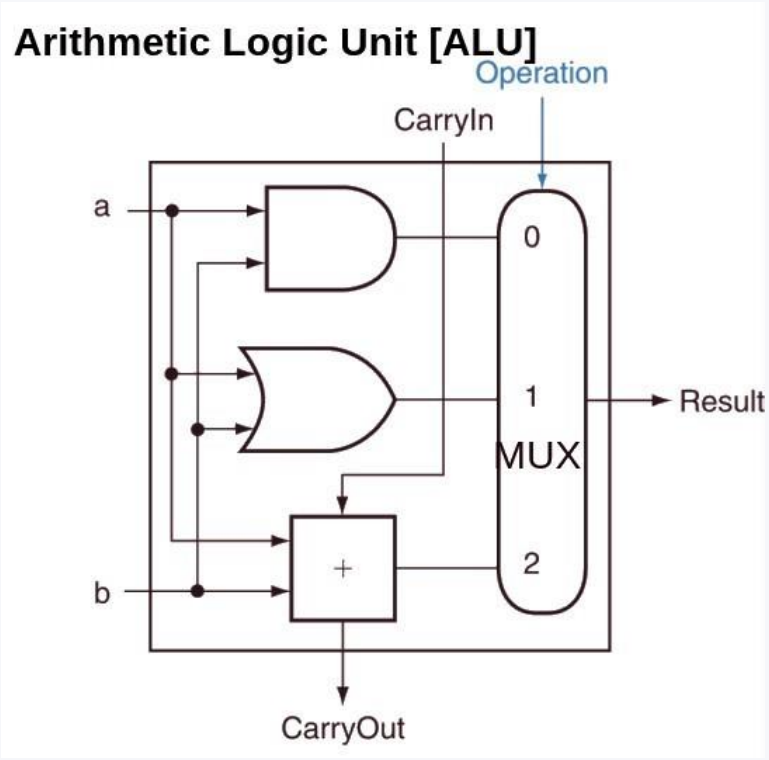
DECIDING PROCESSOR INSTRUCTIONS

- The ALU has only three operations: AND, OR, +
- Thus we don't yet know how many registers in the register bank, we don't yet need that.
- Let's just say there are 4 registers called R1, R2, R3, R4 these could link to a, b or Result
- …



DECIDING PROCESSOR INSTRUCTIONS

- We can then decide for this ISA, we have three instructions, that we can name:
  AND, OR and ADD
- Since they draw data from or move data to one of the registers, the instructions could be represented as:

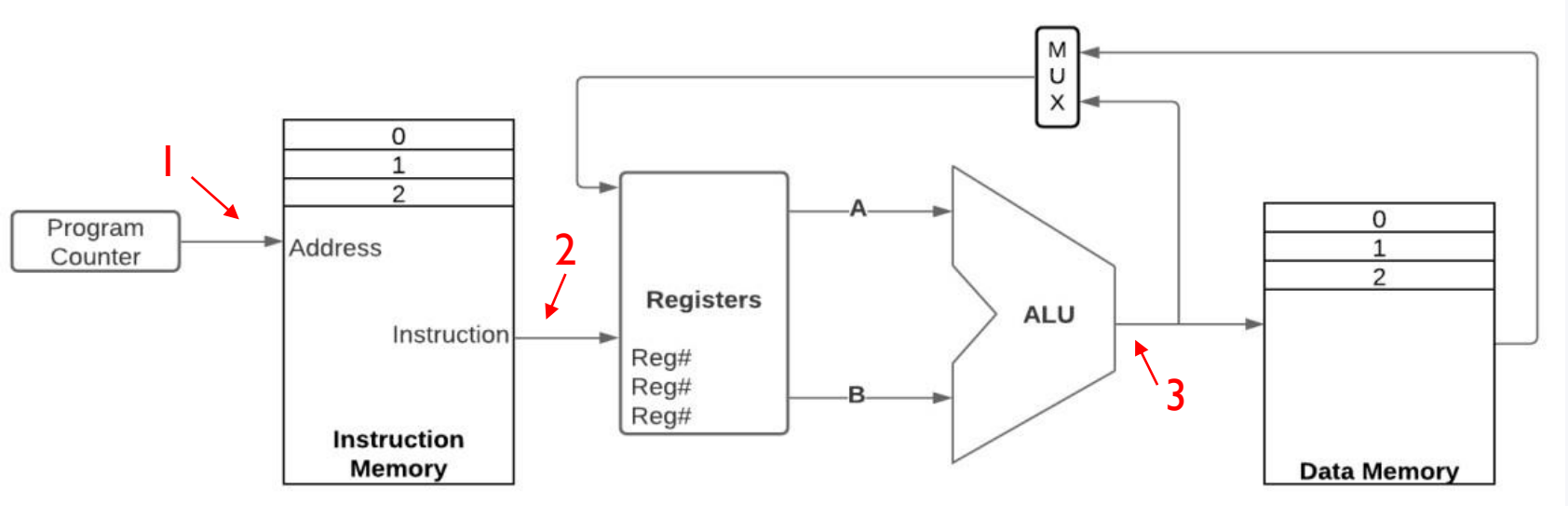



```
AND R1, R2, R3 // R1 = R2&R3
OR  R1, R2, R3 // R1 = R2|R3
ADD R1, R2, R3 // R1 = R2+R3
```

CONNECTING THE PARTS

- The program, comprising instructions to run, is stored in the memory
- The PC (program counter) points to an address in memory where the next instruction is fetched
- The control unit coordinates the fetching of the next instruction from memory, and then selects source data to applied to the operation, and where the result of the operation is sent. (see next slide)

BASIC PROCESSOR OVERVIEW



1. Program Counter points to the Address of an Instruction in the Instruction Memory // using Modified Harvard Architecture in this example*
2. Instruction determines what values are placed on A & B operands for ALU
3. Output from the ALU stored in the Data Memory OR written back into a registers in the register bank

DIFFERENT TYPES OF INSTRUCTIONS NEEDED

- Remember from Lecture L12 regards ARM instructions and their grouping?
- We can utilize the same sort of approach to decide on instructions the CPU should have

BRANCH instructions

J *address* @ jump to address

DATA PROCESSING instructions

ADD R1,R2,R3 @ R1 = R2 + R3

CMP R1,R2 @ compare R1 and R2

SWP R1,R2 @ swap value of R1 and R2

STATUS instructions

MSR @ move status register flags

LOAD and STORE instructions

LDR R1, address @ load from memory

STR R1, address @ store to memory

EXCEPTION instructions

SWI @ cause software interrupt

THINK ABOUT IT!

- You know by now a reasonable bit of assembly programming
- You know the essential parts of a processor
- You know how the parts are coordinated and how data is transferred between them
- You know how instructions refer to a processor task (e.g. ADD R1,R2,R3) that activates a whole lot of parts of the processor doing things (activating circuits and moving data about) …
- So, you hopefully have a view on both the programmer (processor user) and design side of a processor

PLANS FOR LAST WEEK

- Focus on completion of the mini-project
- Study for exams

End of EEE3096/5 Lectures!

Good luck with the exams!
And onwards to a good vacation!