FIT1047 - Week 3

Central Processing Units, Part 2

MONASH University

Recap

In the previous lecture we saw

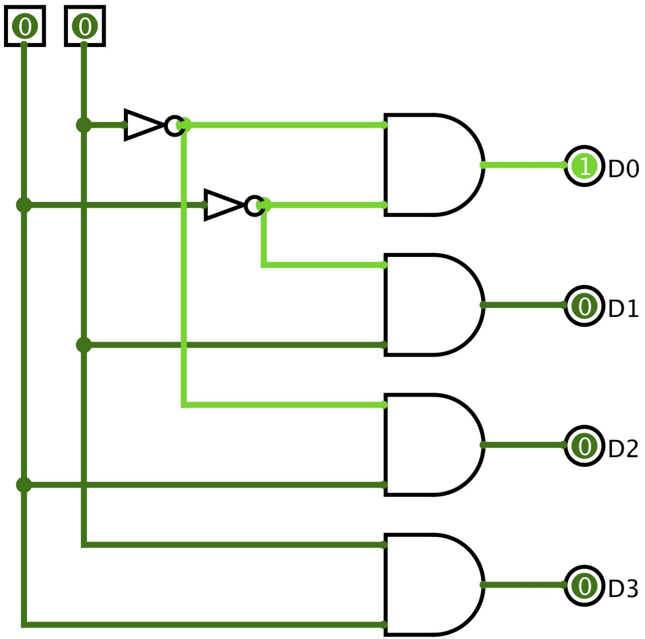
- Basic CPU architecture
- MARIE assembly code
- Combinational circuits (in particular: adders)

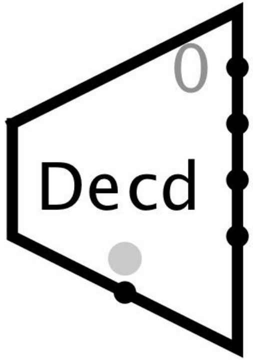
Overview

- Arithmetic / Logic Units (ALUs)
- Sequential circuits
  - Flip flops, registers, counters
  - memory
- Control
  - Executing a program

Decoders

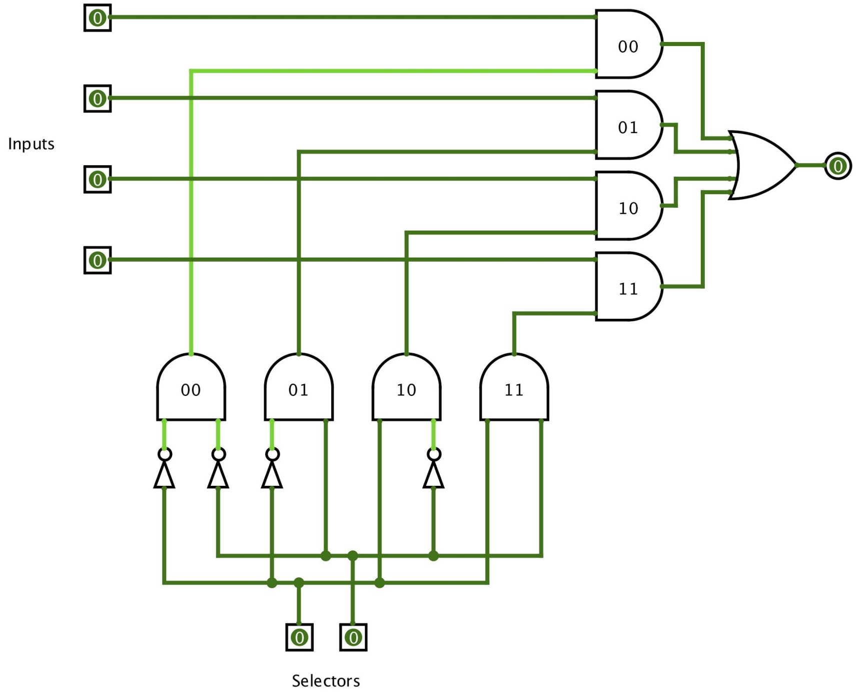
Activate one output based on a binary number

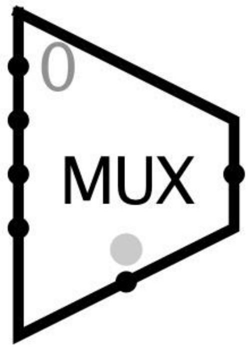




Multiplexers

Select one of several inputs





Arithmetic Logic Unit
(ALU)

ALU

Implements basic computations:

- Integer addition, subtraction (in more complex CPUs: multiplication)
- Comparisons
- Bitwise Boolean operations (AND, OR, NOT)
- Shifting

Inputs:

- Two *n*-bit **operands**
- Op-code (determines the operation to perform)

Outpus:

- *n*-bit result and status flags (overflow? error?)

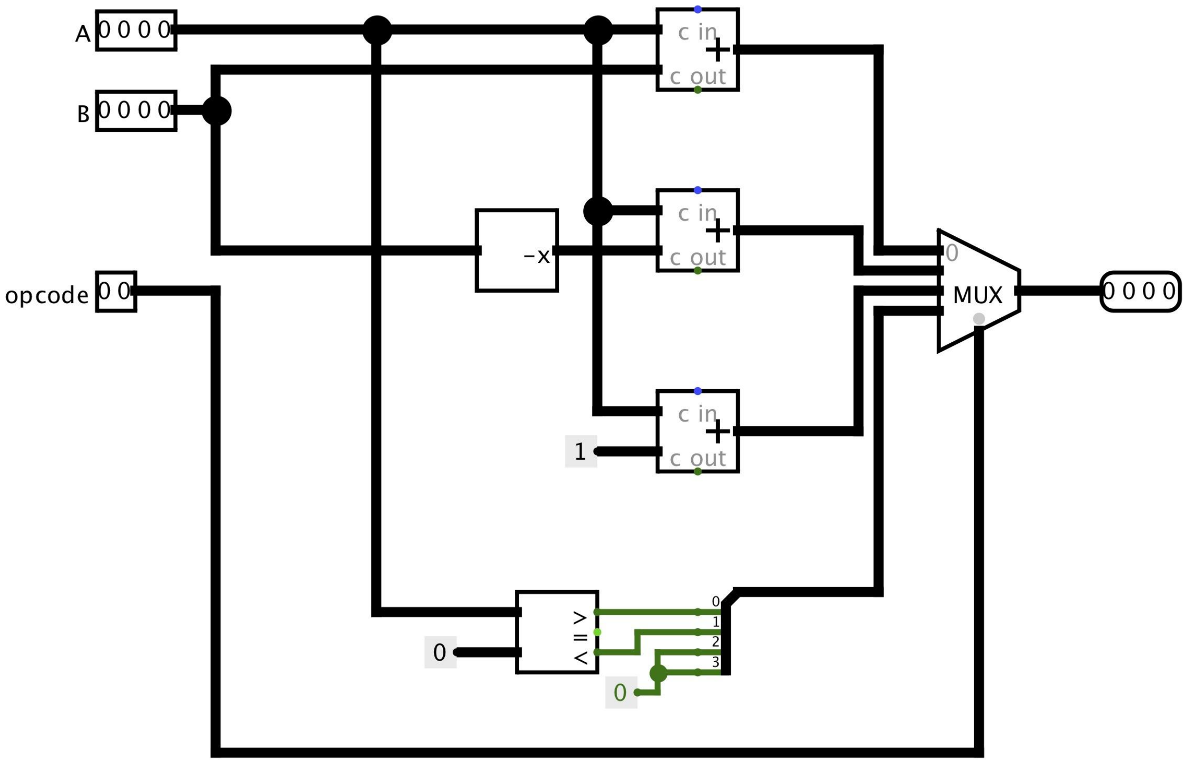
ALU

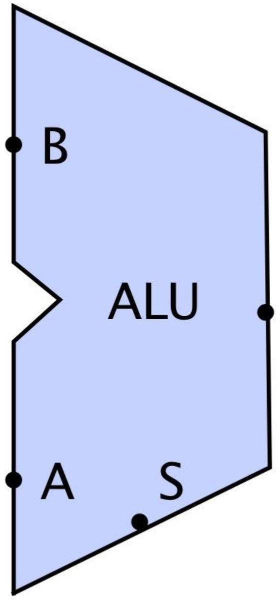
How does the circuit decide which operation to perform?

- Simply do all in parallel
- Then choose the result prescribed by the op-code

Sounds like a job for a MUX!

ALU





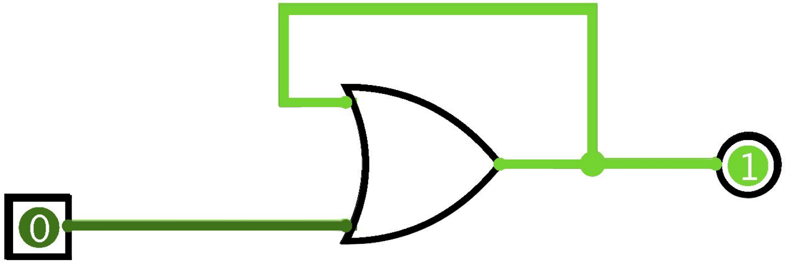
Sequential Circuits

(output depends on *sequence* of inputs)

Sequences

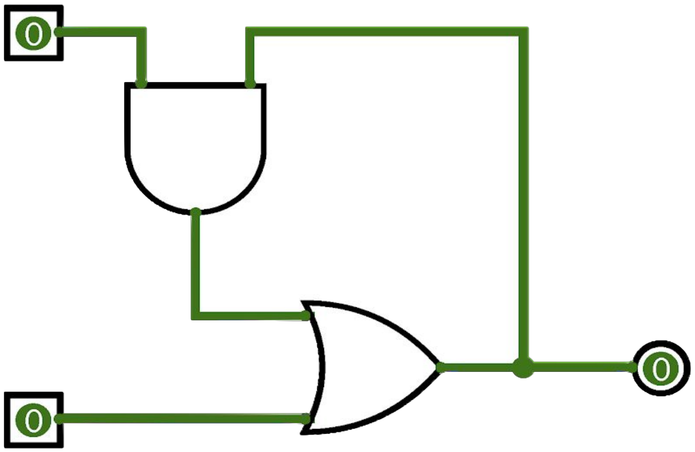
How can a circuit “remember” the past?

Feed the output back into the input!

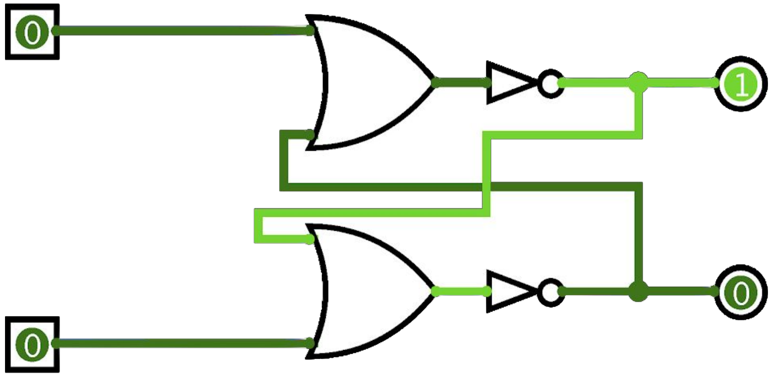


Sequences

Toggle using another input



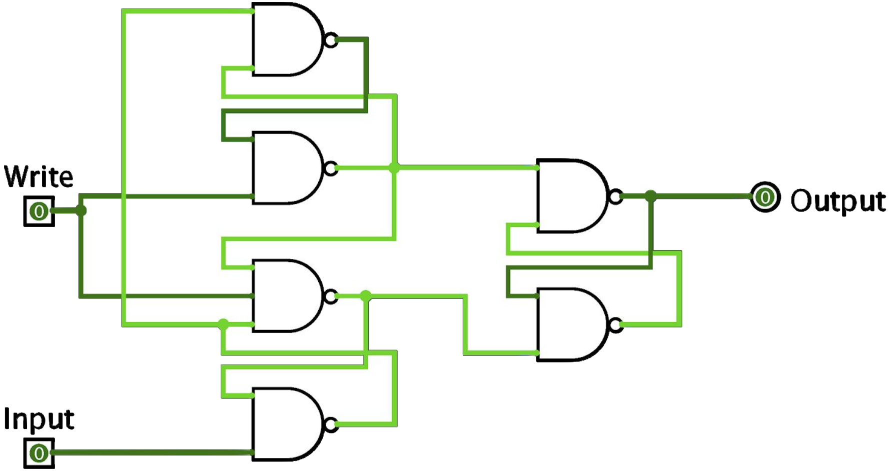
Set/reset latch



But digital circuits use a single bit for data!

D flip-flop

- Two inputs:
  - The bit to be stored
  - A signal: read or write mode
- One output:
  - The bit currently stored



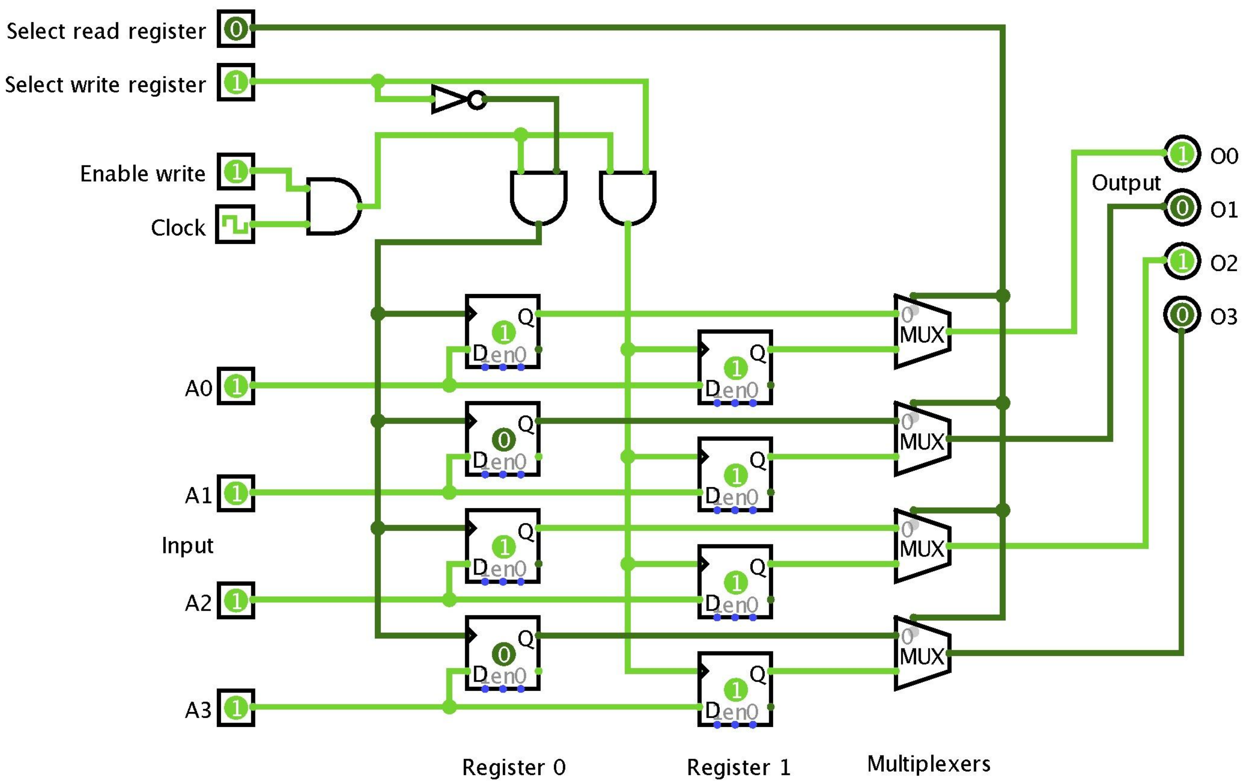
Registers

- Very fast memory inside the CPU
- Some special purpose registers
  - PC, IR, MBR, MAR (for MARIE)
- Some general purpose registers
  - AC (MARIE), AH/AL, BH/BL, CH/CL, DH/DL (x86)
- Fixed bit width
  - E.g. 16 bit in MARIE
  - 16/32 or 64 bits in modern processors

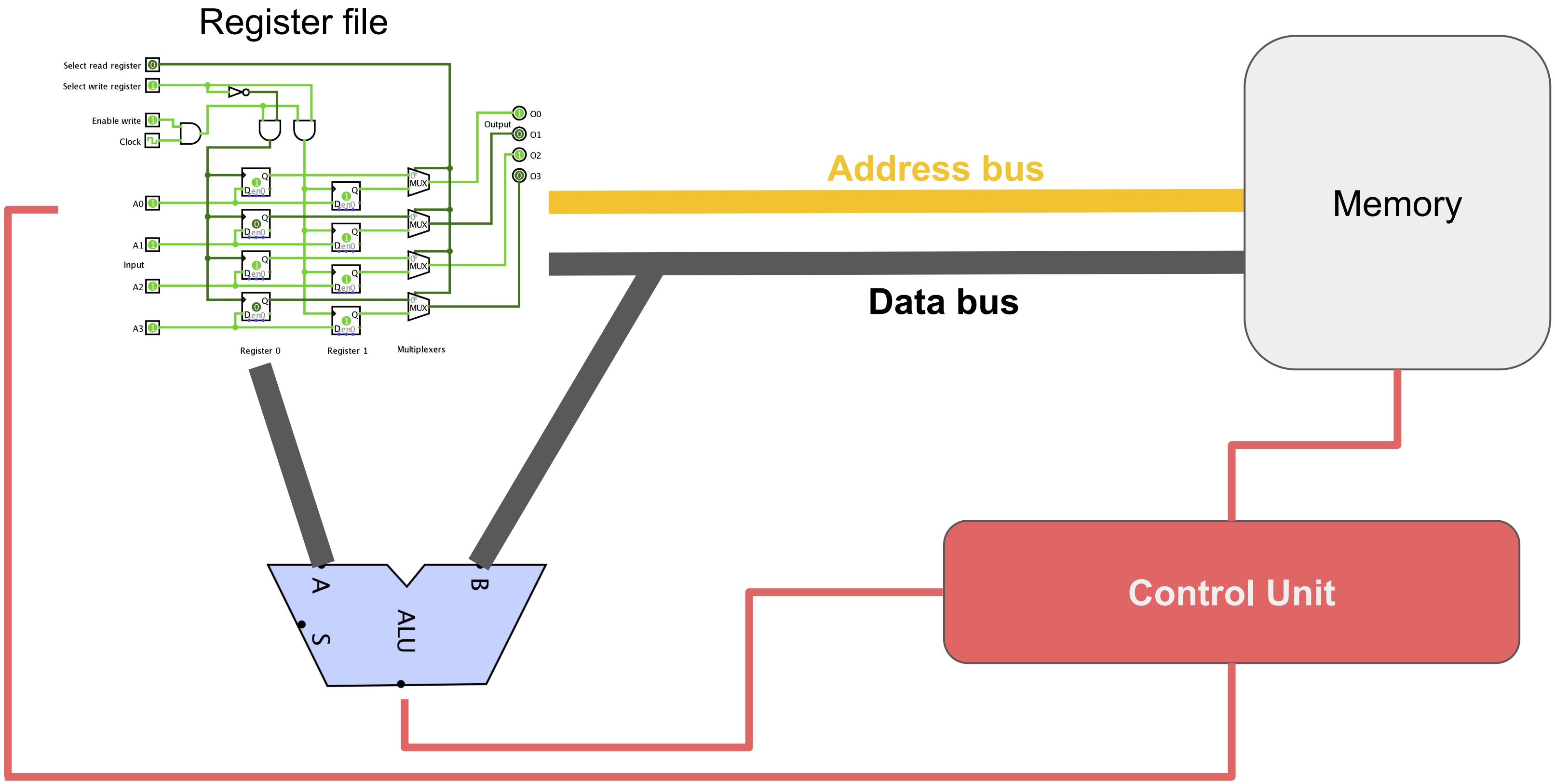
Register file

- Collection of registers
- Each implemented using *n* flip-flops (for *n* bits)
- *n* inputs and outputs
- Additional input: which register to write to
- Additional input: which register to read from

Register file



MARIE architecture



Control Unit

- Controls fetch-decode-execute cycle
- Switches *control signals* on and off:
  - Each signal is a “wire” inside the CPU
  - Which register to read/write
  - Which memory address to read/write
  - Which operation to perform in the ALU
- Needs to “know” which signals to switch on/off for each instruction

Let’s specify, for each instruction, what to do!

Register Transfer Language (RTL)

- Break down instructions into small steps
- CPU performs one step per *clock cycle*
- Each step transfers data between registers and/or memory

RTL: fetch

1. **MAR $\leftarrow$ PC** *load PC into memory address register*
2. **MBR $\leftarrow$ M[MAR]** *load value from memory (M) into memory buffer*
3. **IR $\leftarrow$ MBR** *load value from MBR into instruction register*
4. **PC $\leftarrow$ PC+1** *increment PC to point at next instruction*

RTL: decode

5. **MAR** ← **X** *load address X from IR into MAR*
6. **MBR** ← **M[MAR]** *load value from memory into MBR*

Some instruction need both of these steps, some just step 5, some instructions need neither 5 nor 6.

RTL: execute

Depends on the concrete instruction (of course).

Example: `Add X`

7. **AC ← AC + MBR** *(place result of addition in AC)*

Example: `Jump X` (does not need decode step 6)

6. **PC ← MAR** *(load X, stored in MAR, into PC)*

RTL: Full Add X instruction

1. MAR ← PC
2. MBR ← M[MAR]
3. IR ← MBR
4. PC ← PC+1
5. MAR ← X
6. MBR ← M[MAR]
7. AC ← AC + MBR

Control Signals

Each RTL step tells us which control signals to switch on and off.

Example: MBR $\leftarrow$ M[MAR]

(load memory value from address stored in MAR into MBR)

- Switch register file to write into MBR
- Switch memory into read mode

Control Signals

Each RTL step tells us which control signals to switch on and off.

Example: AC $\leftarrow$ AC + MBR

(add value in MBR to value stored in AC, store result in AC)

- Switch register file to read from AC
- Switch register file to write into AC
- Switch ALU into “Add” mode (ALU always reads one operand from MBR)

Outlook

Tutorials this week:

- MARIE programming
- Circuits for adding and subtracting

Next lecture:

- More MARIE instructions
- Memory
- Input/Output and Interrupts