EECS 151 Disc 8

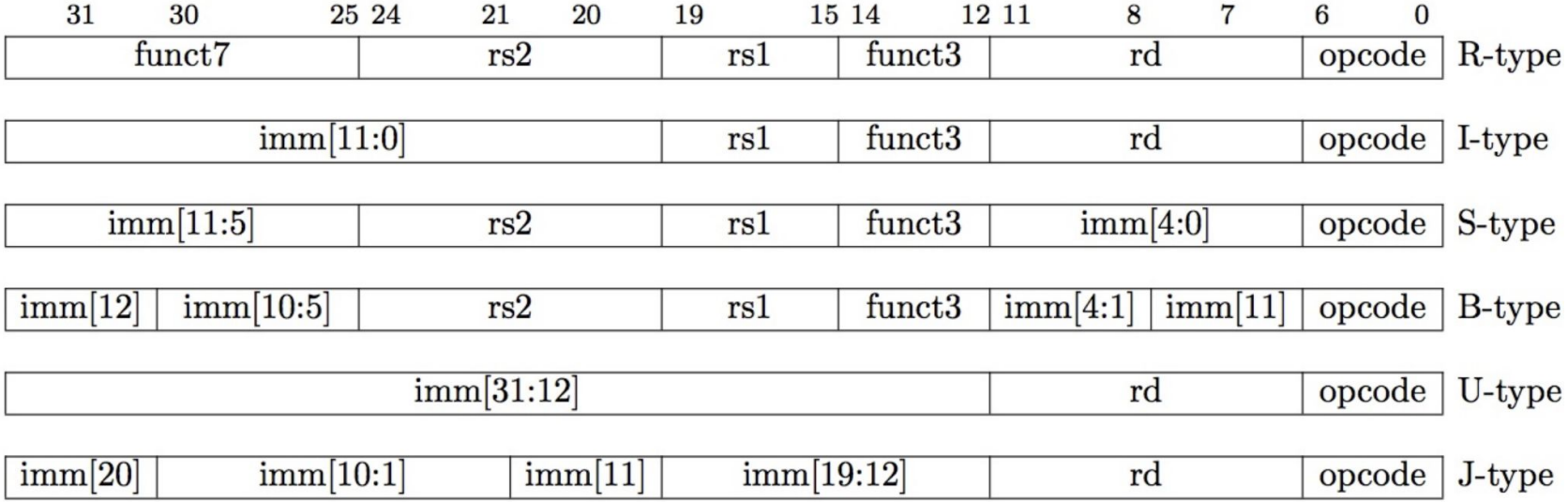
Rahul Kumar (session 1)
Yukio Miyasaka (session 2)

Berkeley
UNIVERSITY OF CALIFORNIA

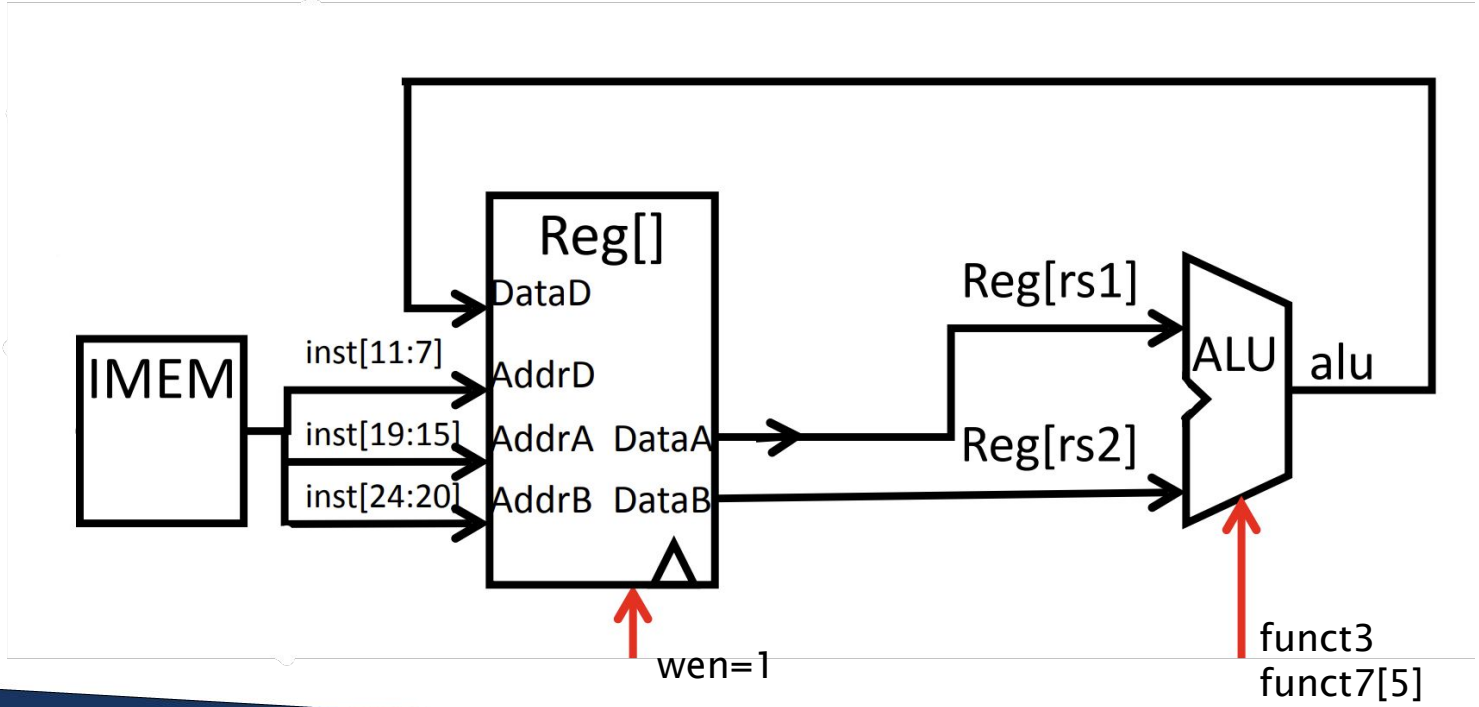
Contents

- RISC-V
- Memory

Instruction Types



Example: R-type Instruction



Design Guide

- (I-type) Add MUXes for the second input of ALU (imm)
  - Generate immediate from instruction
  - Byte-select and sign-extend the loaded data
- (S-type) Connect rs2 to memory input with a shifter
- (B-type) Add a branch comparator and a MUX for the first input of ALU (pc)
  - Add a MUX for next pc
- (U-type) Add a MUX to store immediate in register file
- (J-type) Add a MUX to store pc in register file
  - Add another MUX for next PC
- Add CSR and a MUX to write either rs1 or immediate

Hazards

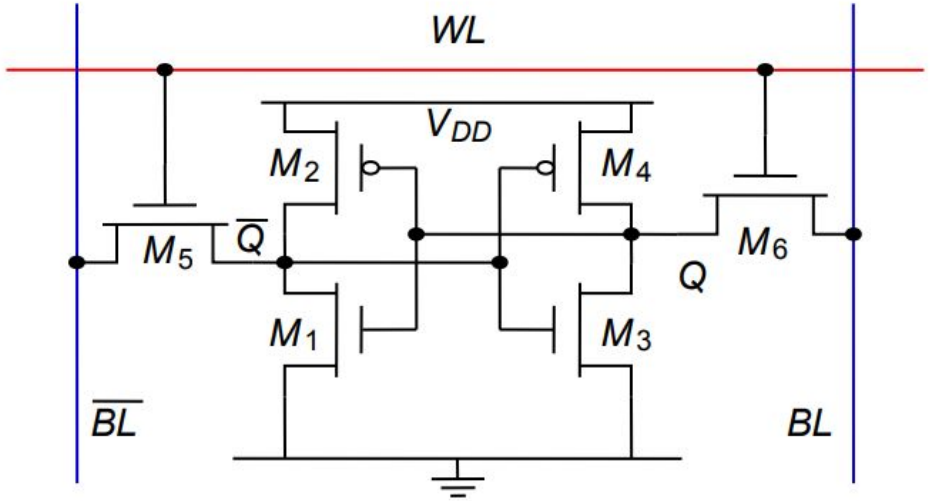
- Data hazards
  - Using ALU output, memory output, immediate, or pc to be stored in register file in the next instruction (and the next next instruction if register read is not in the EX stage)
  - Stall and inject NOPs in the EX stage
  - Or forward those values from pipeline registers (and memory output if it is not the critical path)
- Control hazards
  - We don't know which Instruction to execute after a branch or jump instruction
  - Stall pc until the next instruction address is determined while injecting NOPs
  - Forwarding may help a little (setting the memory address and next pc at the same time)
  - Predict (e.g. always not taken) and flush wrong instructions when mispredict

Iron law

Execution time = #instructions * CPI * Clock period

- #instructions is fixed for each program if using the same ISA and compiler
- CPI (Cycles per instruction) is amortized number of cycles per instruction
  - Inverse of number of instruction completed per cycle
- Clock period is decided by the critical path i.e. the longest pipeline stage
- Optimization is to decrease CPI * Clock period at a reasonable cost

SRAM

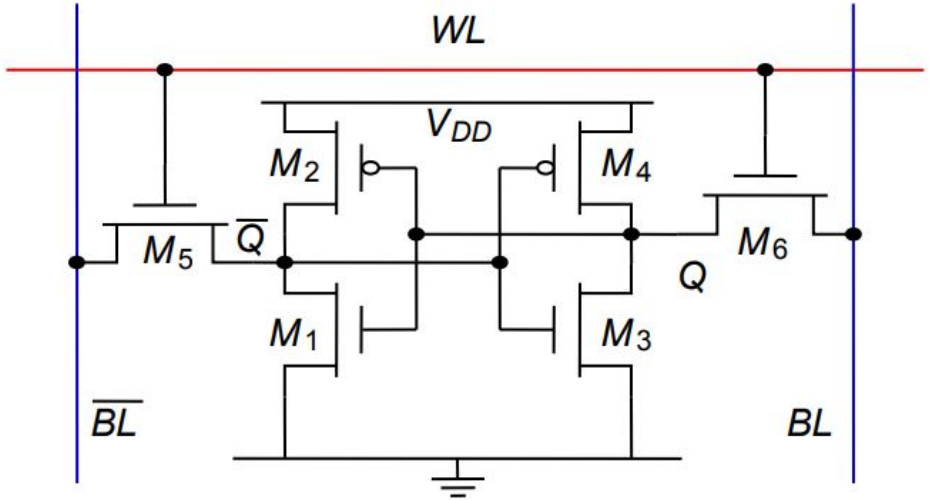


SRAM: Read

Charged to Vdd

Charged to Vdd

Enable is on if we want to read this row



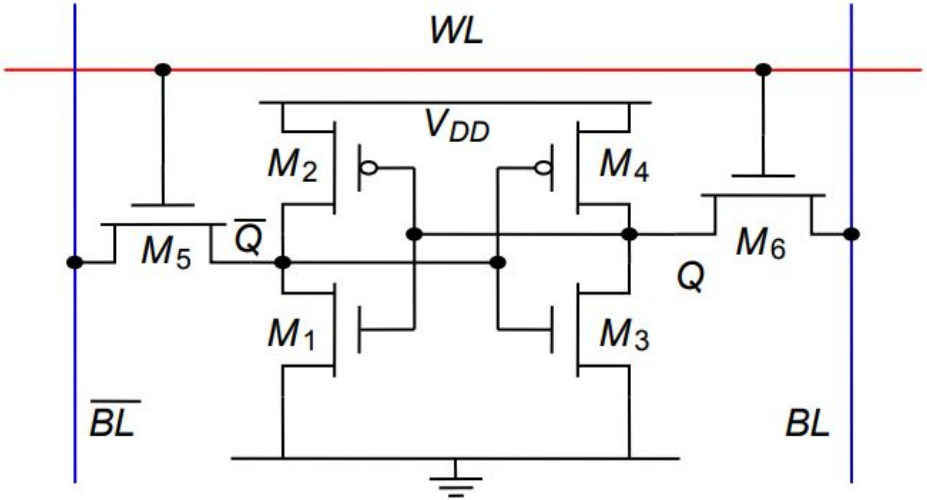
Either left or right line is discharged through M1 or M3
(M5 and M6 should have larger resistance than M1 and M3)

SRAM: Write

Connected to X’

Connected to X

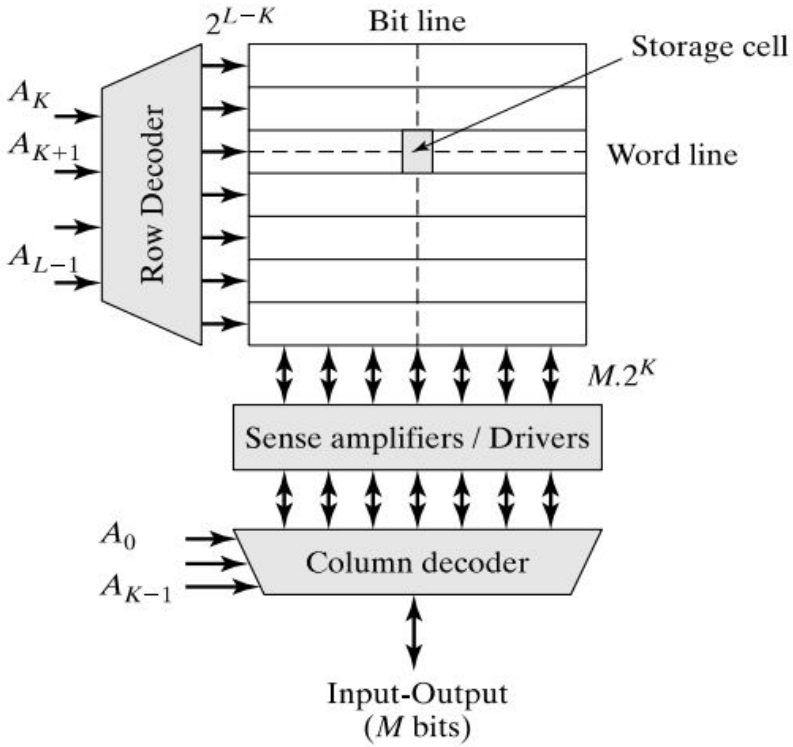
Enable is on if we want to write this row



Either left or right line is set to GND
Overwrite the output of inverter to GND through M2 or M4
(M5 and M6 should have smaller resistance than M2 and M4)

Memory

- log2(#rows) bits are given to select one row
- log2(#units per row) bits are given to select one unit
  - Unit may be Byte or Word (4 Bytes)
- Make sure log2(#units in memory) bits are given in total
- Decoder to enable one row
- Another decoder may be needed to extract one unit from the row



Predecoder

- N-bit AND gates are sometimes too big (for N-bit decoder)
- Use multiple levels of smaller AND gates while sharing fanins

Example: Two-Level 8-bit Decoder

- Use 4-input AND gates in the first level, and 2-input AND gates in the second level (we could divide in a different way e.g. 2-input then 4-input)
- Requires (2 * 2^4) 4-input AND gates and (2^8) 2-input AND gates
- Smaller than (2^8) 8-input AND gates (count number of transistors for AND gates) (#inverters is 8 regardless #levels)

First level

```
X0 = a'b'c'd',    Y0 = e'f'g'h',
X1 = a'b'c'd ,    Y1 = e'f'g'h ,
X2 = a'b'c d',    Y2 = e'f'g h',
X3 = a'b'c d ,    Y3 = e'f'g h ,
X4 = a'b c'd',    Y4 = e'f g'h',
…,                …;
```

Second level

```
Z0 = X0 Y0,
Z1 = X0 Y1,
Z2 = X0 Y2,
…,
Z16 = X1 Y0,
Z17 = X1 Y1,
…;
```

Example: Three-Level 8-bit Decoder

- Use 2-input AND gates in each level
- Requires (4 * 2^2) 2-input AND gates in the first level, (2 * 2^4) in the second level, and (2^8) gates in the last level
- Even smaller than the two-level decoder

First level		Second level		Third level
A0 = a'b',	C0 = e'f',	X0 = A0 B0,	Y0 = C0 D0,	Z0 = X0 Y0,
A1 = a'b ,	C1 = e'f ,	X1 = A0 B1,	Y1 = C0 D1,	Z1 = X0 Y1,
A2 = a b',	C2 = e f',	X2 = A0 B2,	Y2 = C0 D2,	Z2 = X0 Y2,
A3 = a b ,	C3 = e f ,	X3 = A0 B3,	Y3 = C0 D3,	…,
B0 = c'd',	D0 = g'h',	X4 = A1 B0,	Y4 = C1 D0,	Z16 = X1 Y0,
B1 = c'd ,	D1 = g'h ,	…,	…;	Z17 = X1 Y1,
B2 = c d',	D2 = g h',			…;
B3 = c d ,	D3 = g h ;			