Review Problem 15

- Orange runs at 1GHz, and provides a unit making all floating point operations take 1 cycle. Grape runs at 1.2 GHz by deleting the unit, meaning floating point operations take 20 cycles. Which machine is better?

It depends. How much floating point

Processor Performance Summary

Machine performance:

$$\text{CPU execution time for a program} = \text{Instructions for a program} * \text{CPI} * \frac{1}{\text{Clock rate}}$$

Complex Instr Set computers

Reduced Instr, Set

Better performance:

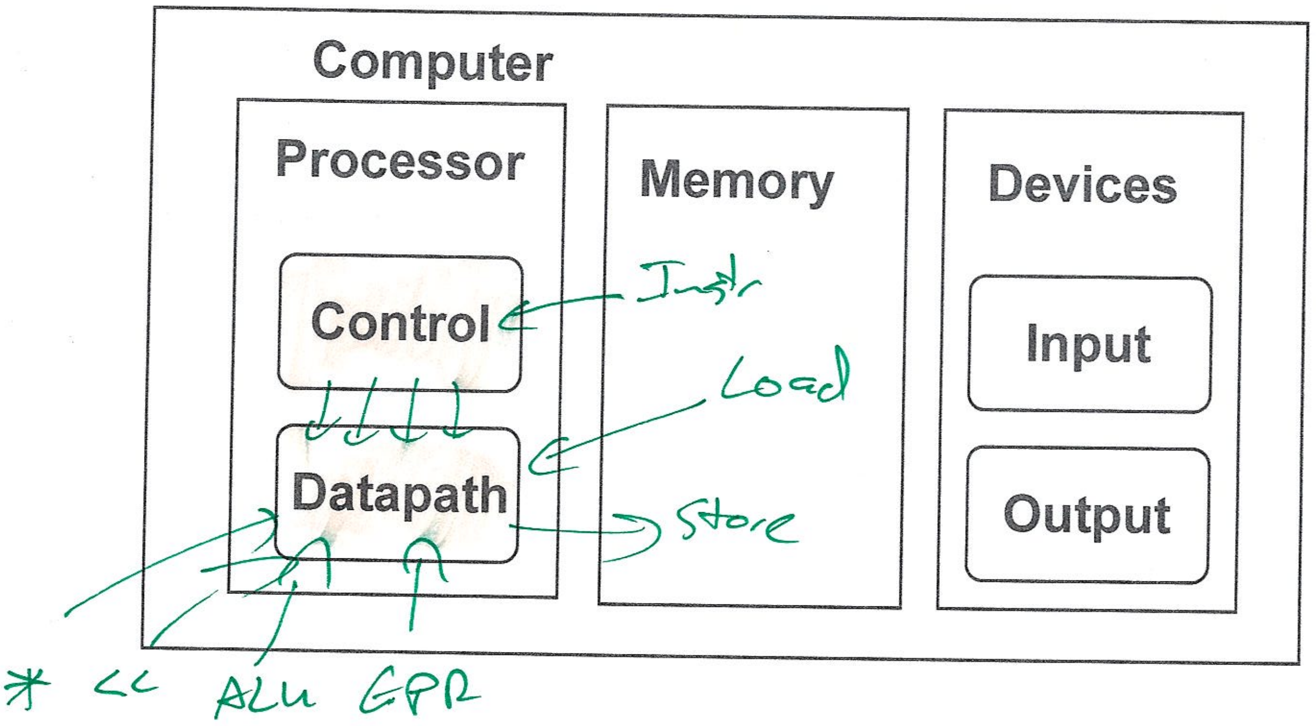
		CISC	RISC
↓	number of instructions to implement computations	↓	↑
↓	CPI	↑	↓
↑	Clock rate	↓	↑
		Intel	

Improving performance must balance each constraint

Example: 1980's RISC vs. CISC

Datapath & Control

Readings: 4.1-4.4



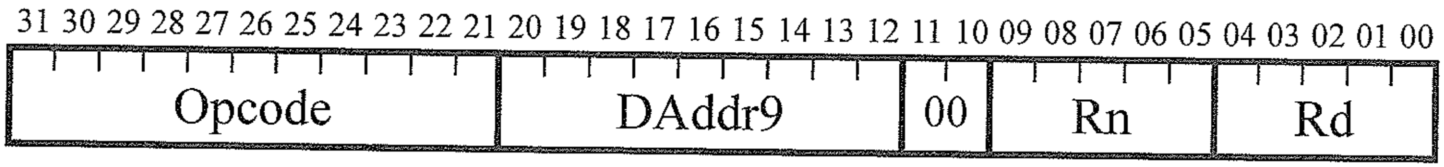
Datapath: System for performing operations on data, plus memory access.

Control: Control the datapath in response to instructions.

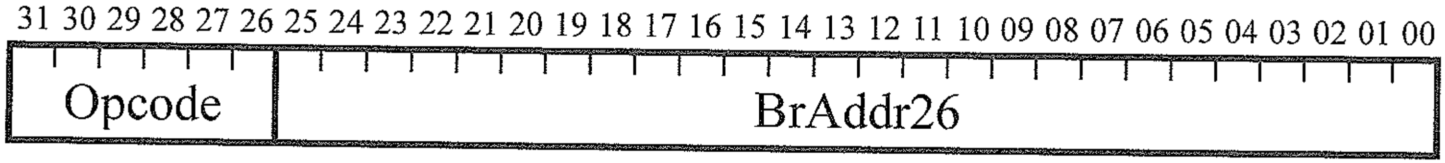
Simple CPU

Develop complete CPU for subset of instruction set

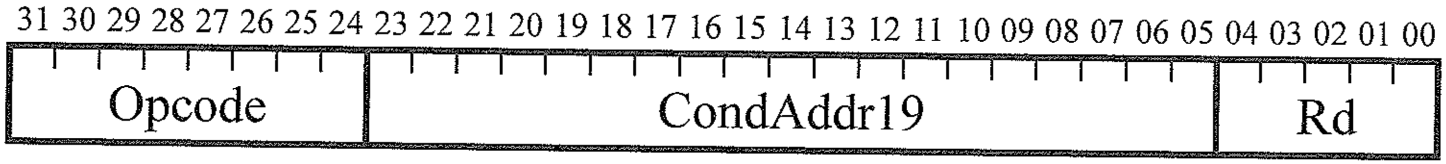
Memory: `LDUR, STUR`



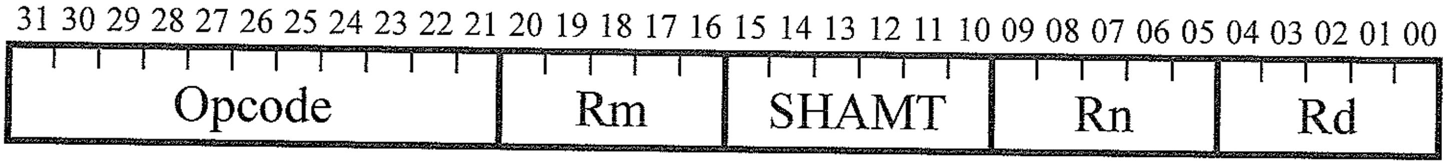
Branch: `B`



Conditional Branch: `CBZ`

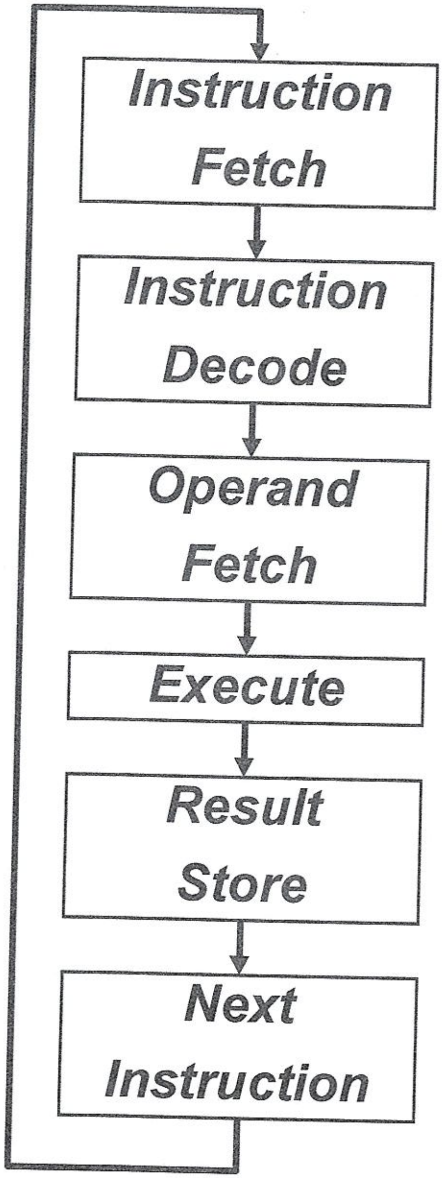


Arithmetic: `ADD, SUB`



Most other instructions similar

Execution Cycle



Obtain instruction from program storage

Read MEM[PC]

Determine required actions and instruction size

Split the instr into field

Locate and obtain operand data

Read Regfile

Compute result value or status

Use ALU/ */<<

Deposit results in storage for later use

Write Regfile

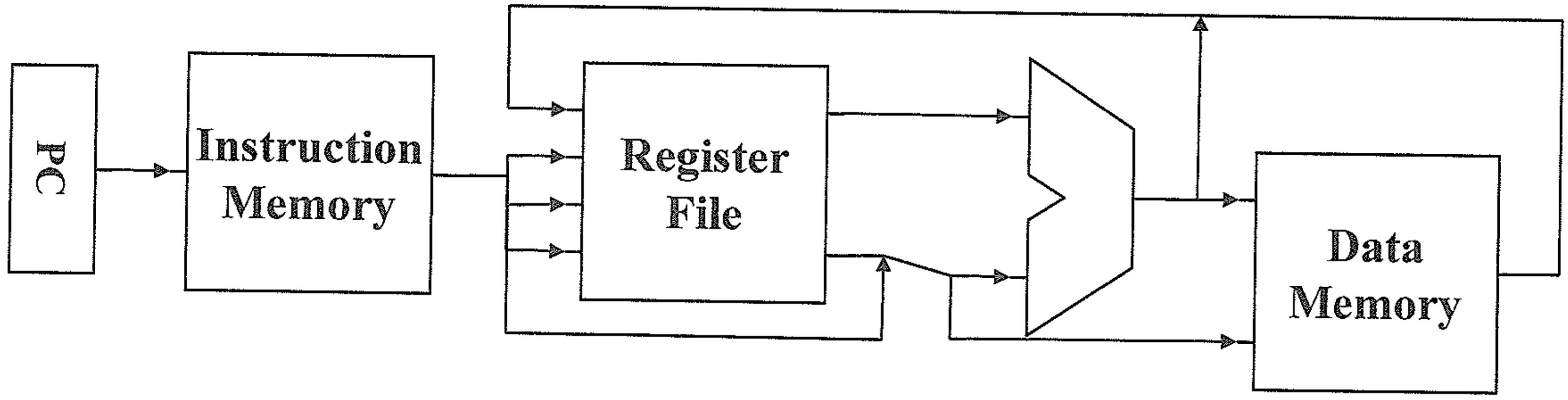
Determine successor instruction

Update PC

Processor Overview

Overall Dataflow

- PC fetches instructions
- Instructions select operand registers, ALU immediate values
- ALU computes values
- Load/Store addresses computed in ALU
- Result goes to register file or Data memory



RTL & Processor Design

Convert instructions to Register Transfer Level (RTL) specification

```
RegA = RegB + RegC;
```

RTL specifies required interconnection of units, control

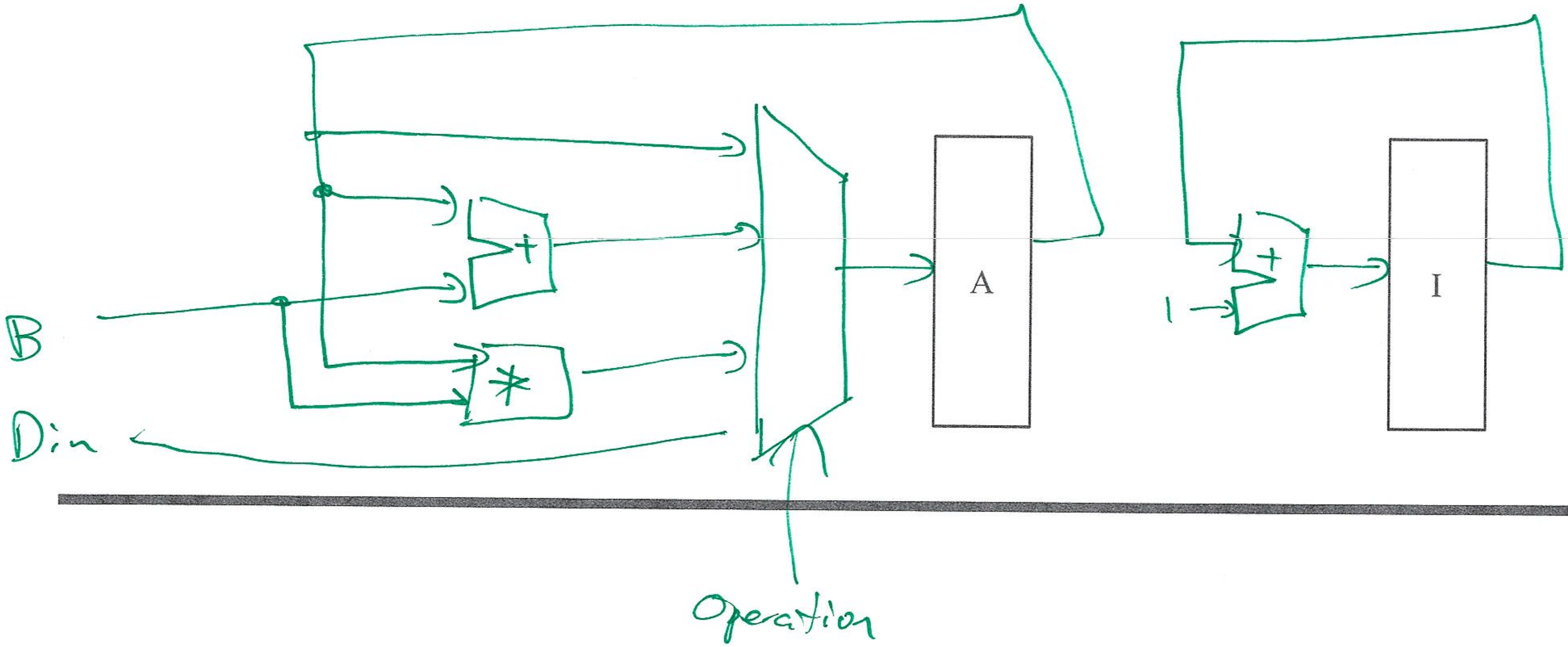
Math unit example:

(add): A = A + B; I++;

(mult): A = A * B: I++;

(hold): A = A; I++;

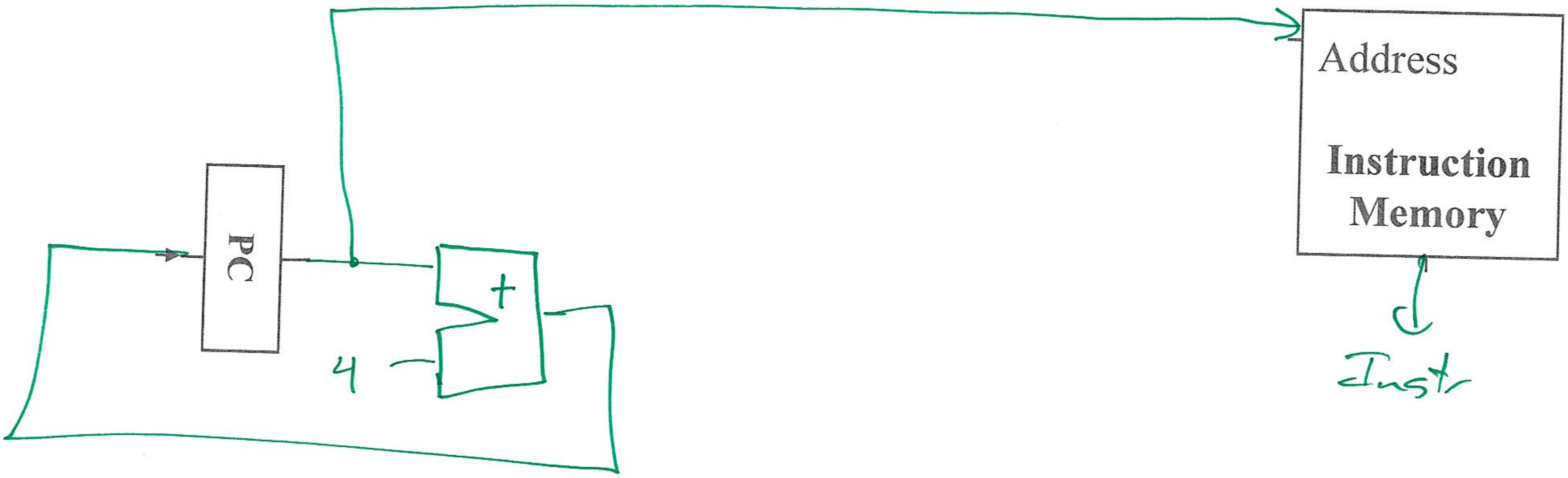
(init): A = Din; I++;



Instruction Fetch

Instr = MEM[PC]

PC = PC+4



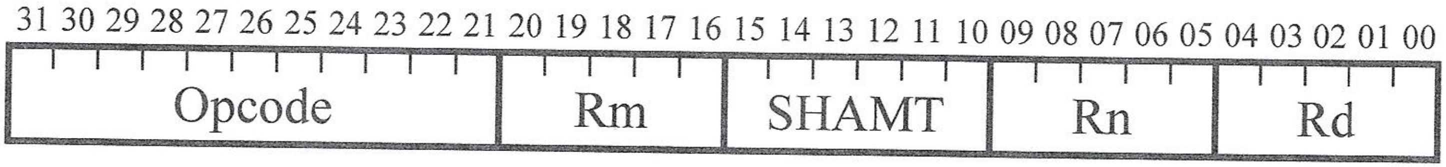
Add/Subtract RTL

Add instruction: `ADD Rd, Rn, Rm`

```
Instr = Mem[PC];
Reg[Rd] = Reg[RN] + Reg[Rm];
PC = PC+4;
```

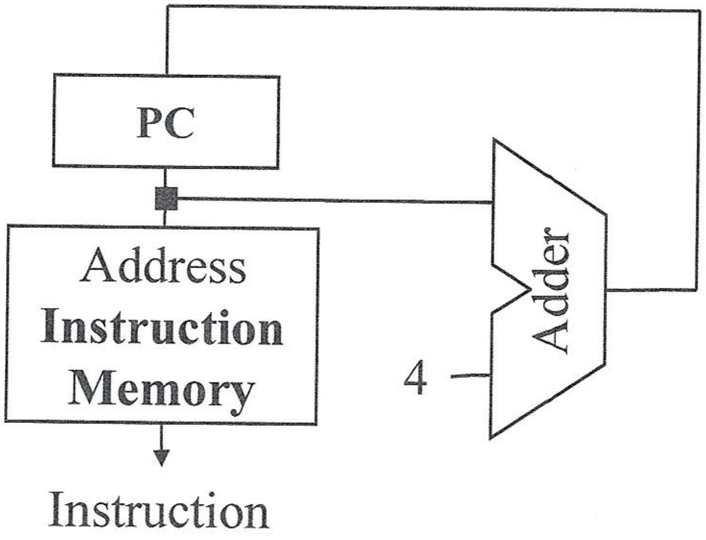
Subtract instruction: `SUB Rd, Rn, Rm`

```
Instr =Mem [PC];
Reg [Rd] = Reg [RN] - Reg [Rm];
PC = PC+4;
```



Add/Subtract Datapath

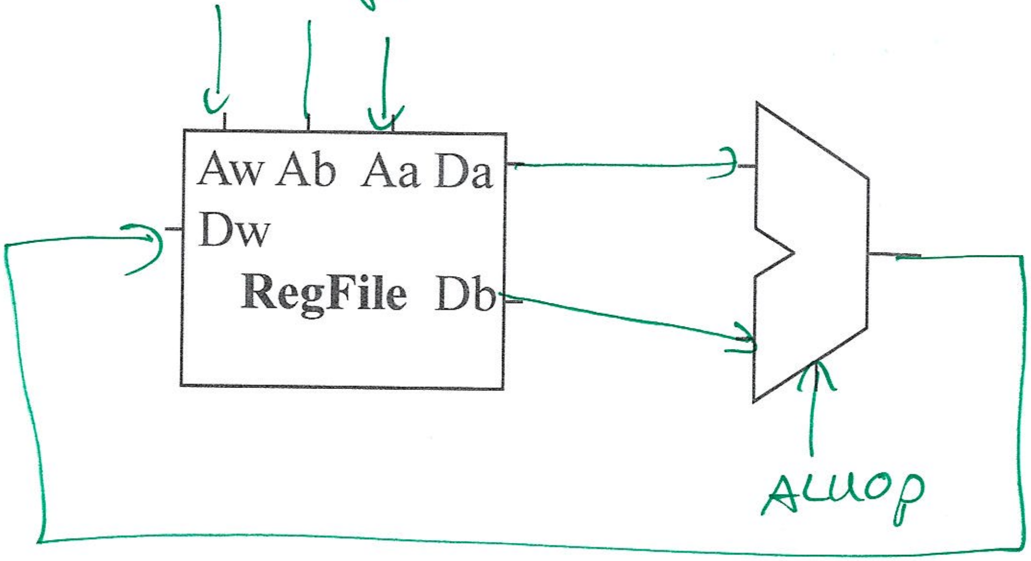
Reg[Rd] = Reg[Rn] op Reg[Rm];



Rd = Inst[4:0]

Rm = Inst,[20:16]

Rn = Inst,[9:5]

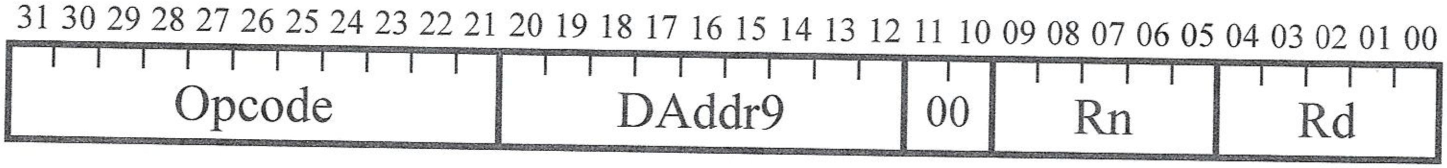


LDUR X4,[X2,#0]

Load RTL

Load Instruction: `LDUR Rd, [Rn, DAddr9]`

```
Instr = Mem[PC];
Addr = Reg[Rn] + SignExtend(DAddr9)
Reg[Rd] = Mem[Addr];
PC = PC+4;
```

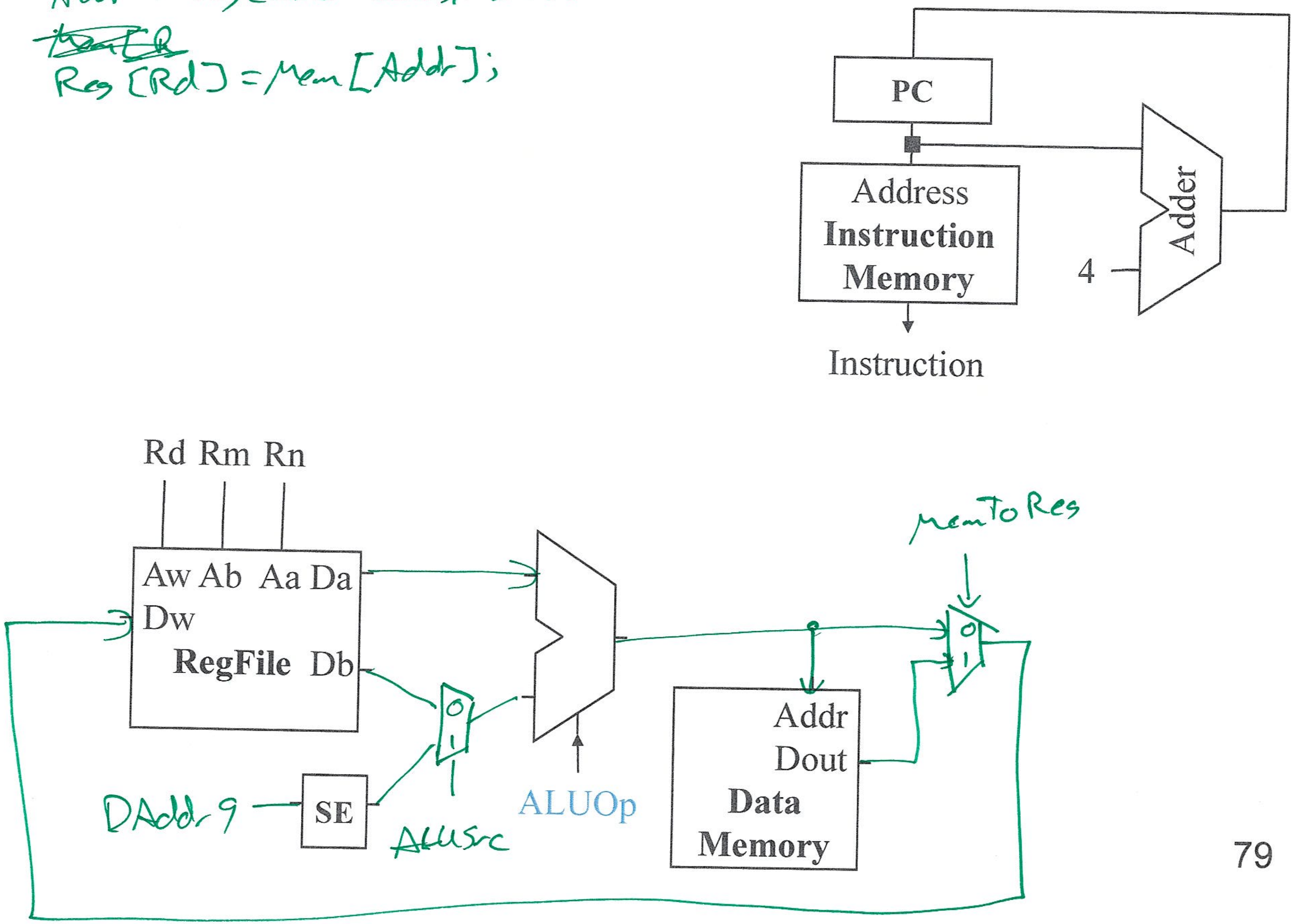


Datapath + Load

Addr = Reg[Rn] + SE(DAddr9);

~~[illegible]~~

Reg[Rd] = Mem[Addr];



Store RTL

Store Instruction: `STUR Rd, [Rn, DAddr9]`

Instr = MEM[PC];

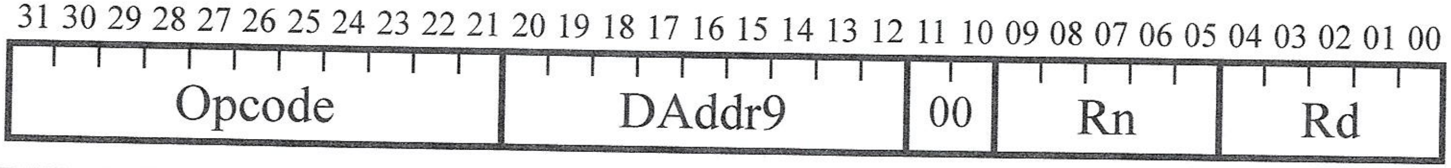
Addr = Reg[Rn] + SE(DAddr9);

~~PC=PC+4~~;

Mem[Addr] = Reg[Rd];

~~PC+~~

PC=PC+4;


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
Opcode
DAddr9
00
Rn
Rd