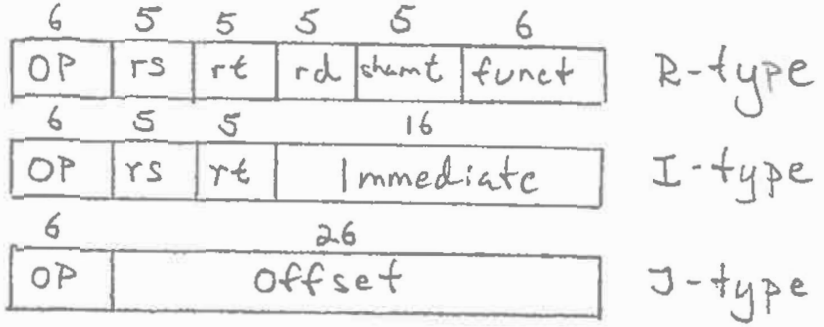
MIPs ISA Summary

(minus floating point)

- 64 bit Architecture
- Register Organization
  - 32 GPRs (each 64 bits)
  - R1 - R31
  - R0 ≡ 0 always
- Data types
  - 8-bit (byte)
  - 16-bit (half-word)
  - 32-bit (word)
  - 64-bit (double word)
- Addressing modes
  - immediate
  - displacement
  - PC-relative
  - register indirect
- Memory
  - byte addressable
  - big or little endian
- Fixed instruction format



MIPs ISA Summary

(minus floating point)

- Arithmetic Ops
  - 3-address reg-reg
    - rd ← rs funct rt (R-type)
  - 3-address reg-imm
    - rt ← rs op Imm (I-type)
- Data transfer Ops
  - 3-address reg-imm
    - rt ← *[rs + Imm] (I-type)
    - *[rs + Imm] ← rt (I-type)
- Control Ops

Conditional branch (I-type)

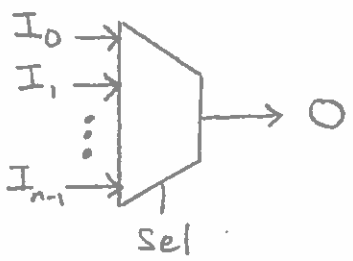
BNEZ {
- IP ← IP+4 if rs == 0
- IP ← IP+4+(Imm<<2) if rs != 0

Unconditional branch

- J: IP ← IP+4+(Offset<<2) (J-type)
- JAL: R31 ← IP+4, IP ← IP+4+(Offset<<2) (J-type)
- JR: IP ← rs (I-type)
- JALR: R31 ← IP+4, IP ← rs (I-type)

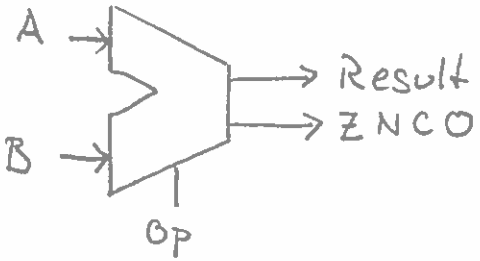
Microarchitecture Building Blocks

Multiplexers



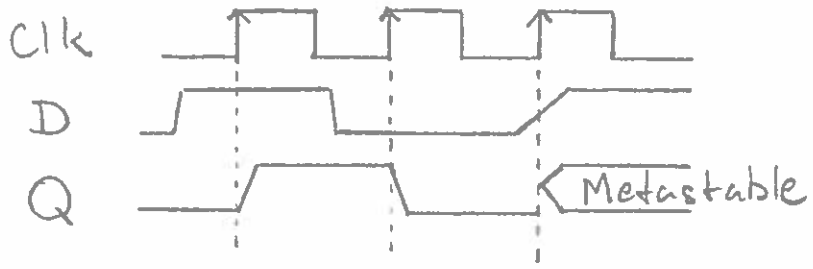
Output follows one of the inputs based on "Sel".

Arithmetic Unit (ALU)



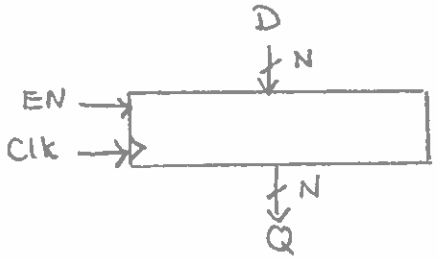
"Op" selects an ALU function
Arithmetic: ADD, SUB, ...
Logical: AND, OR, NOT, SHIFT, ...
Comparison: GT, LT, EQ, ...

Edge-Triggered Flip-Flop



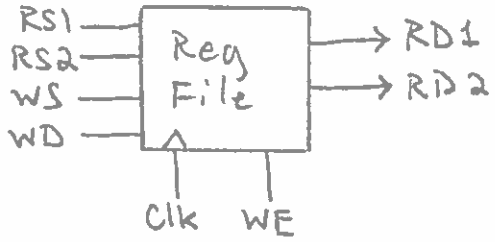
Microarchitecture Building Blocks

Single Register



Output "Q" follows input "D" on clock edge in Enable "EN" asserted.

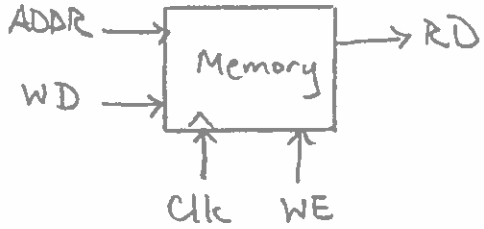
Register File



Read ports: "RD1" + "RD2" controlled by "RS1" + "RS2".

Write port: "WD" controlled by "WD". Writes occur only when enable "WE" asserted.

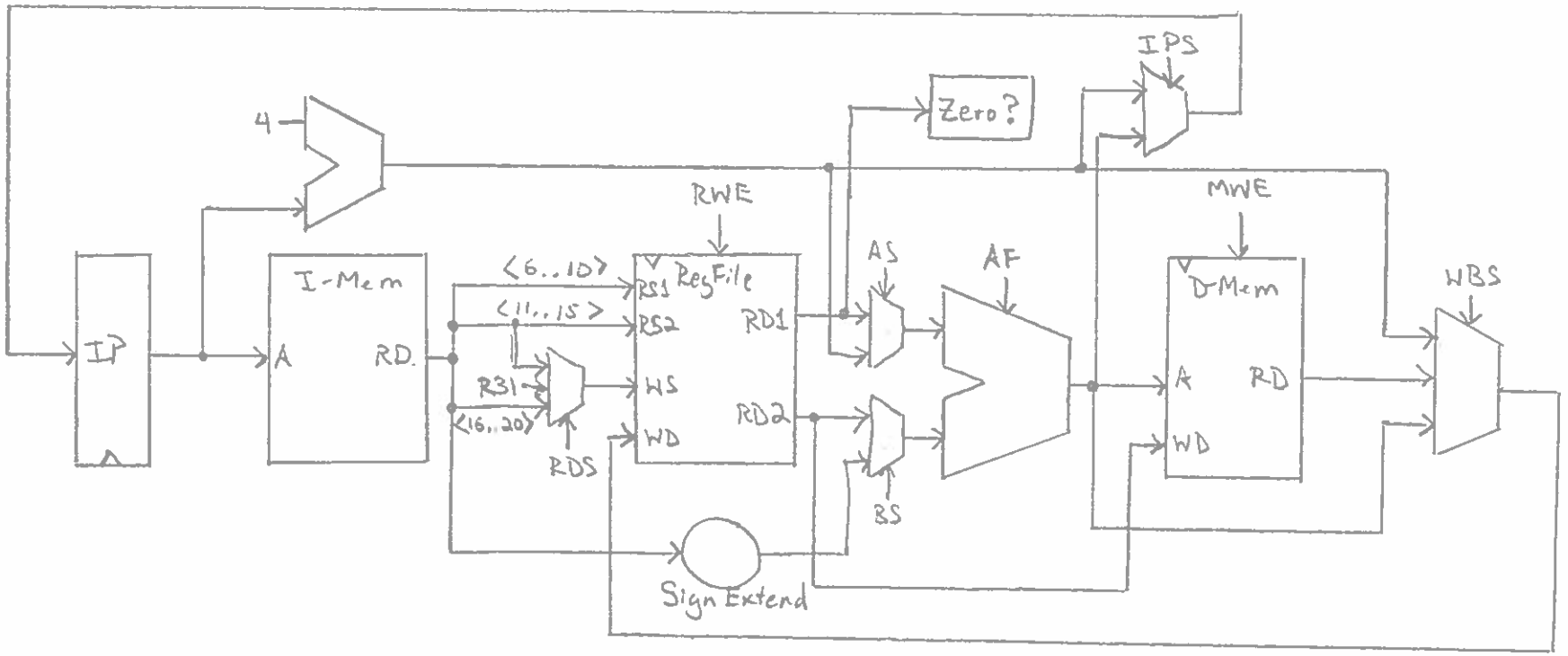
Memory



Read is combinational

Write happens on clock edge if enable "WE" asserted.

Combinational MIPS Implementation



Handwired Control

	RDS	RWE	AS	BS	AF	MWE	WBS	IPS
ALU	rf3 <16..20>	yes	RD1	RD2	FUNC	no	ALU	IP+4
ALUi	rf2 <11..15>	yes	RD1	Imm_{16}	OP	no	ALU	IP+4
LD	rf2 <11..15>	yes	RD1	Imm_{16}	A+B	no	Mem	IP+4
SD	-	no	RD1	Imm_{16}	A+B	yes	-	IP+4
BEQZ	-	no	IP+4	Imm_{16}	A+B	no	-	IP+4 if NZ, ALU if Z
JUMP	-	no	IP+4	Off_{26}	A+B	no	-	ALU
JAL	R31	yes	IP+4	Off_{26}	A+B	no	IP+4	ALU
JR	-	no	RD1	-	A+0	no	-	ALU
JALR	R31	yes	RD1	-	A+0	no	IP+4	ALU

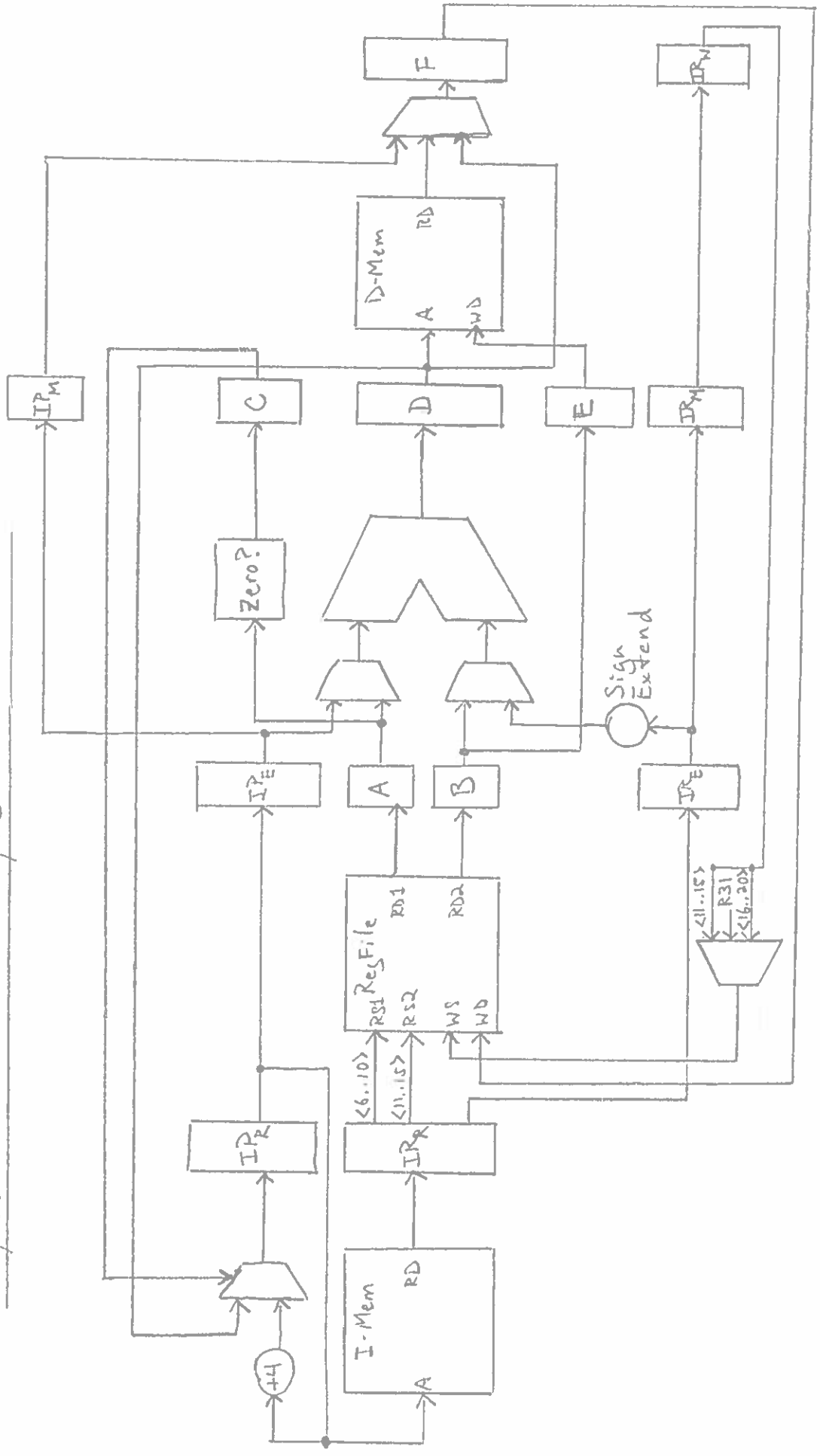
At rising edge of clock:

update IR
update regfile
update data mem

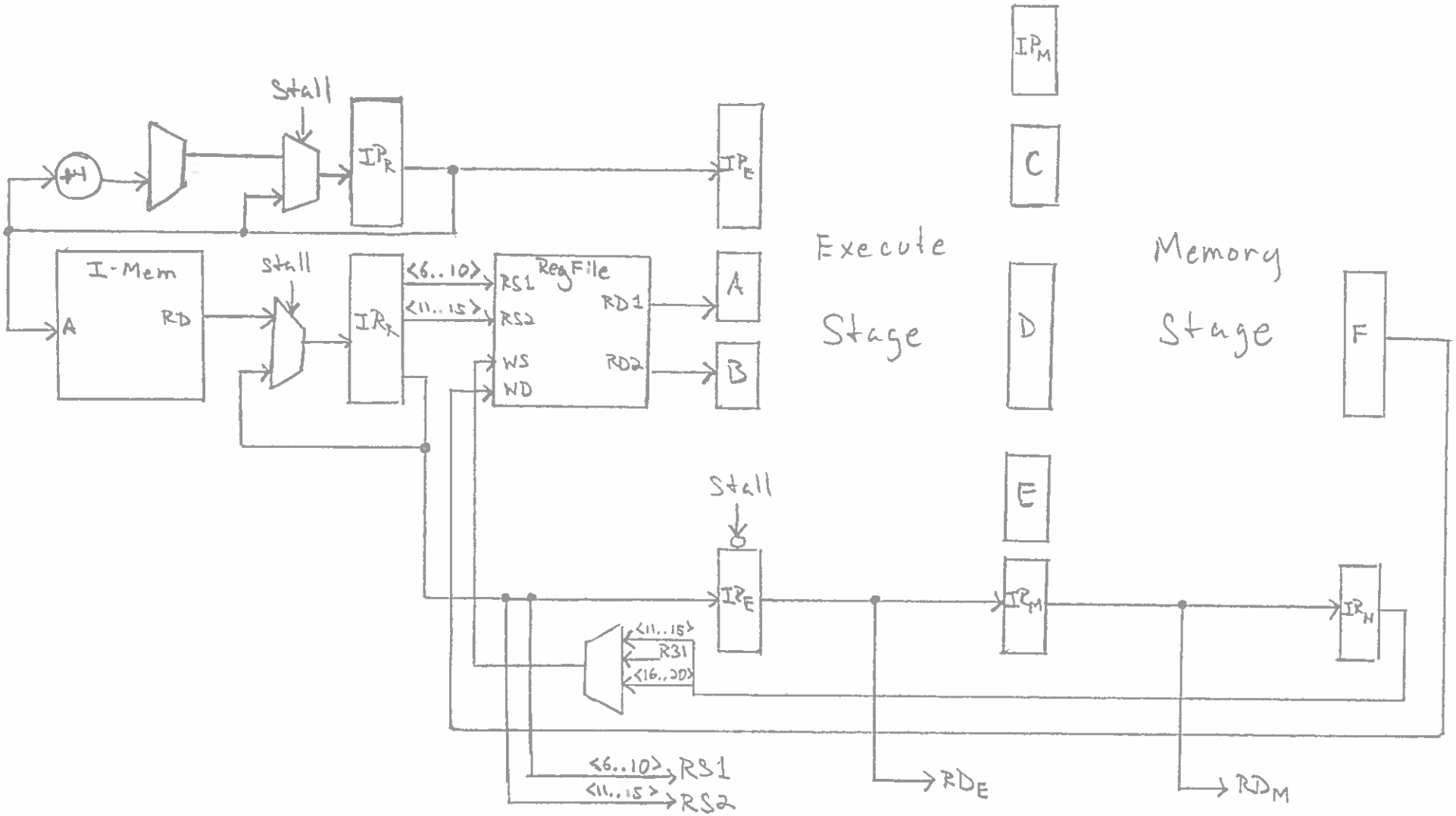
clock timing:

$$t_c > t_{FETCH} + t_{RFETCH} + t_{ALU} + t_{DMEM} + t_{RWB}$$

Pipelined MIPS Implementation



Stalling Pipeline for Data Hazards



Stall Equation

Register Use

	source	dest
ALU	rf1, rf2	rf3
ALUi	rf1	rf2
LD	rf1	rf2
SD	rf1, rf2	-
B__Z	rf1	-
J	-	-
JAL	-	31
JR	rf1	-
JALR	rf1	31

Qualifying terms

$$RD = \begin{cases} rf3, & \text{if opcode == ALU} \\ rf2, & \text{if opcode == ALUi, LD} \\ 31, & \text{if opcode == JAL, JALR} \end{cases}$$

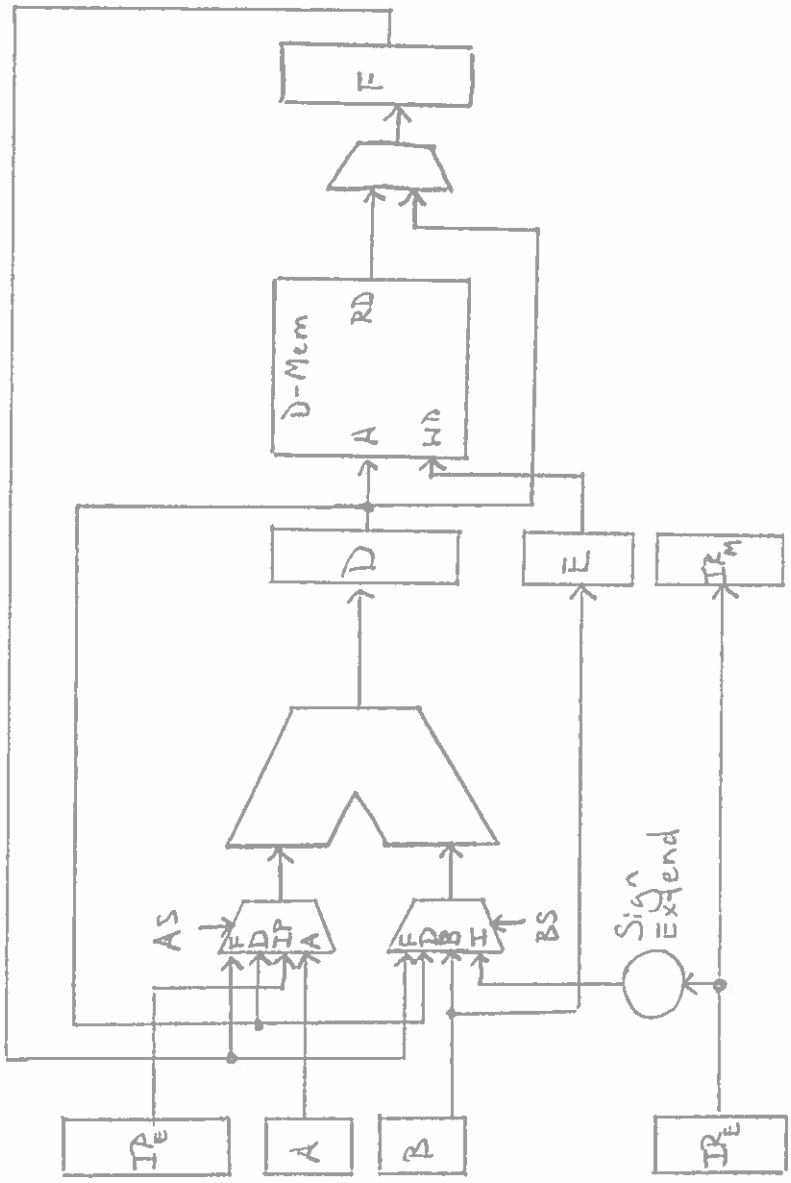
$$we = \begin{cases} (RD \neq 0), & \text{if opcode == ALU, ALUi, LD, JAL, JALR} \\ \text{false}, & \text{otherwise} \end{cases}$$

$$re1 = \begin{cases} \text{true}, & \text{if opcode == ALU, ALUi, LD, SD, BZ, JR, JALR} \\ \text{false}, & \text{if opcode == J, JAL} \end{cases}$$

$$re2 = \begin{cases} \text{true}, & \text{if opcode == ALU, SD} \\ \text{false}, & \text{otherwise} \end{cases}$$

$$\begin{aligned} Stall = ((&rs_R == RD_E) \cdot we_E + \\ &(rs_R == RD_M) \cdot we_M) \cdot re1_R + \\ ((&rt_R == RD_E) \cdot we_E + \\ &(rt_R == RD_M) \cdot we_M) \cdot re2_R \end{aligned}$$

Bypassing to the ALU



Decode Stage