CS 451 / 551 / ECE 541

ADVANCED COMPUTER ARCHITECTURE

SESSION no. 6

University of Idaho

APPENDIX C: INSTRUCTION PIPELINING

INSTRUCTION LEVEL PARALLELISM (ILP)

- TRANSPARENT TO PROGRAMMER (COMPILER MAY CHANGE)

ASSEMBLY LINE

~~CISC~~ - SIMPLE INSTRUCTIONS EASIER RISC TO PIPELINE

- BREAK INSTRUCTION ~~EXECE~~ EXECUTION INTO STAGES
  - IMPLEMENT EACH STAGE IN SEPARATE ~~HARDWARE~~
  - SIMULTANOUSLY

University of Idaho

. INCREASE THRUPUT.

MIPS MODEL

MORE THAN 1 CLOCK/INSTRUCTION (CPI)

~~DATA~~

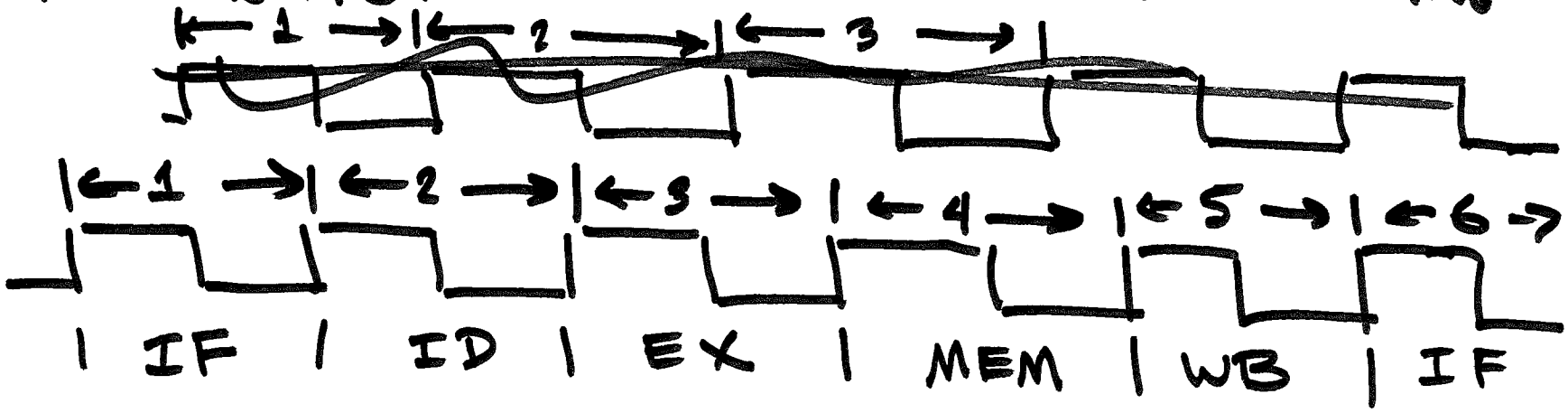
DATA TRANSFER

ARITHMETIC/LOGICAL

CONTROL

~~FLOATING PT~~

EXECUTION STEPS - NO PIPELINING



1. IR ← MEM [PC]
   PC ← PC+1 . next instr.
2. ID
   DATA TRANSFER - COMPUTE ADDR
   ALU - READ DATA REGISTERS
   BR - EQUALITY TEST; COMPUTE BRANCH TARGET

3. EX -

MEM - Compute mem addr (use ALU)

ALU

DADD

REG →
REG → ALU → VALUE

REG →
IMMED → ALU → VALUE

4. MEM - read or write to address computed in EX

5. WB - ALU VALUE → REG

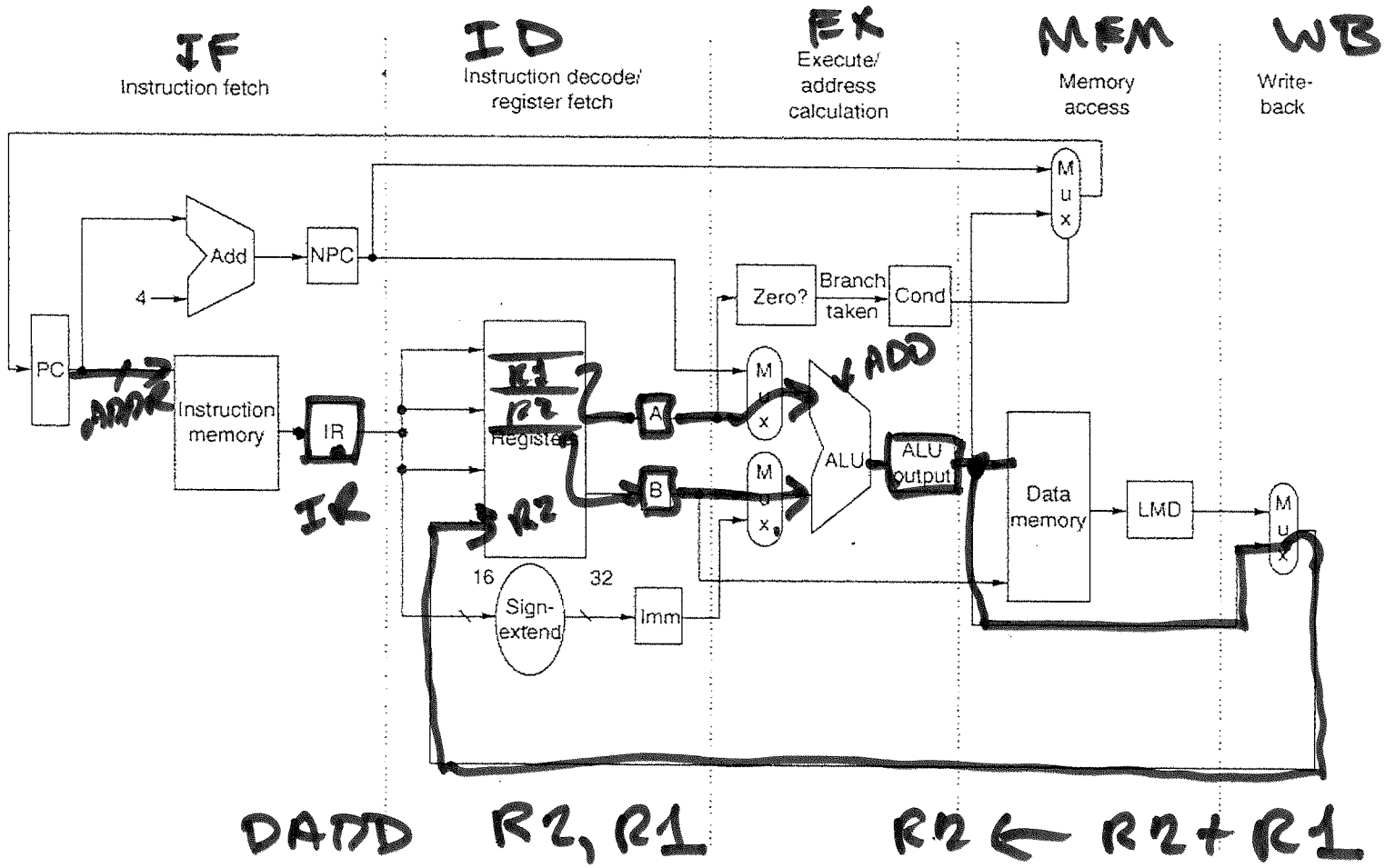


Figure C.21 The implementation of the MIPS data path allows every instruction to be executed in 4 or 5 clock cycles. Although the PC is shown in the portion of the data path that is used in instruction fetch and the registers are shown in the portion of the data path that is used in instruction decode/register fetch, both of these functional units are read as well as written by an instruction. Although we show these functional units in the cycle corresponding to where they are read, the PC is written during the memory access clock cycle and the registers are written during the write-back clock cycle. In both cases, the writes in later pipe stages are indicated by the multiplexer output (in memory access or write-back), which carries a value back to the PC or registers. These backward-flowing signals introduce much of the complexity of pipelining, since they indicate the possibility of hazards.

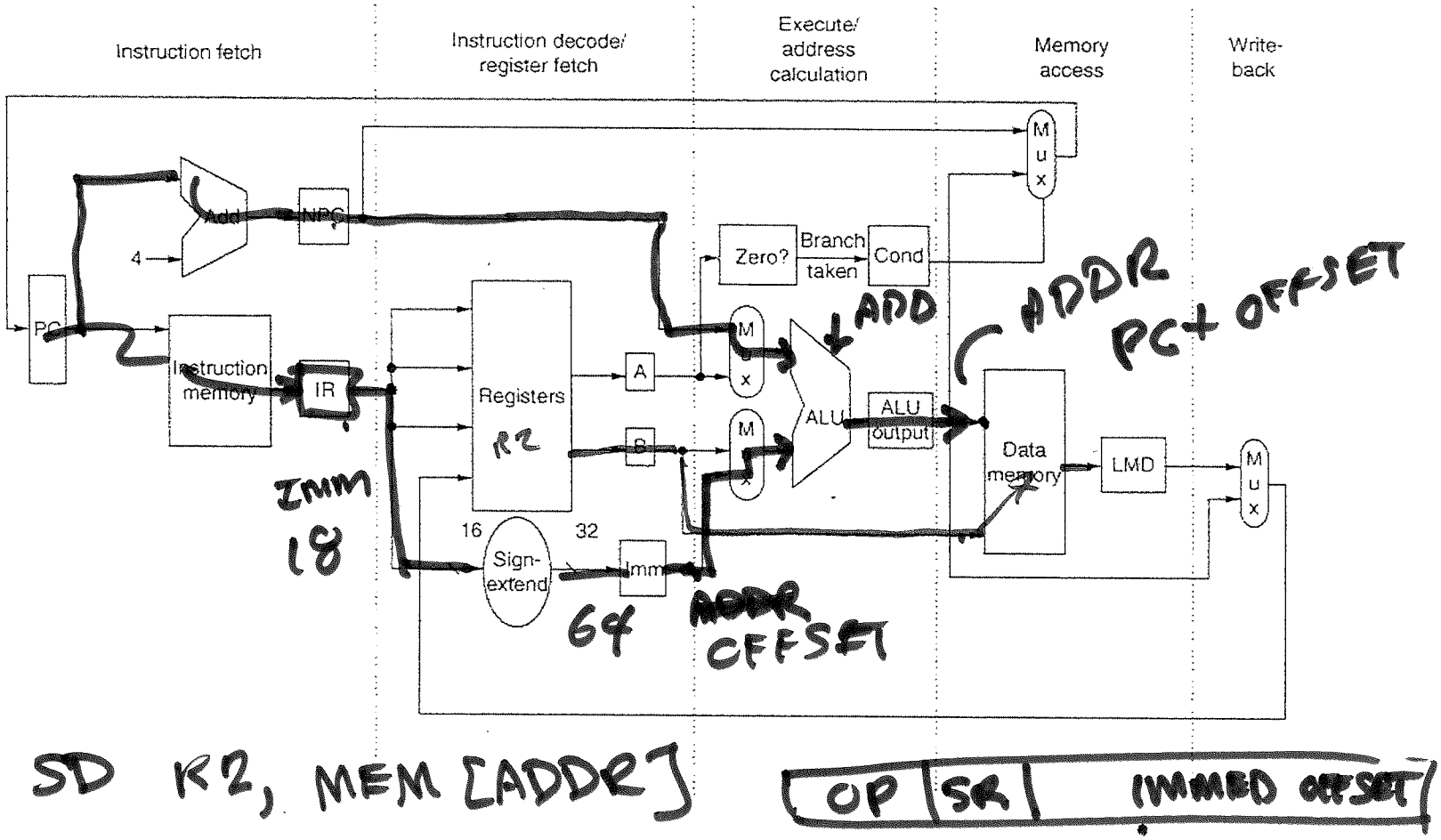


Figure C.21 The implementation of the MIPS data path allows every instruction to be executed in 4 or 5 clock cycles. Although the PC is shown in the portion of the data path that is used in instruction fetch and the registers are shown in the portion of the data path that is used in instruction decode/register fetch, both of these functional units are read as well as written by an instruction. Although we show these functional units in the cycle corresponding to where they are read, the PC is written during the memory access clock cycle and the registers are written during the write-back clock cycle. In both cases, the writes in later pipe stages are indicated by the multiplexer output (in memory access or write-back), which carries a value back to the PC or registers. These backward-flowing signals introduce much of the complexity of pipelining, since they indicate the possibility of hazards.

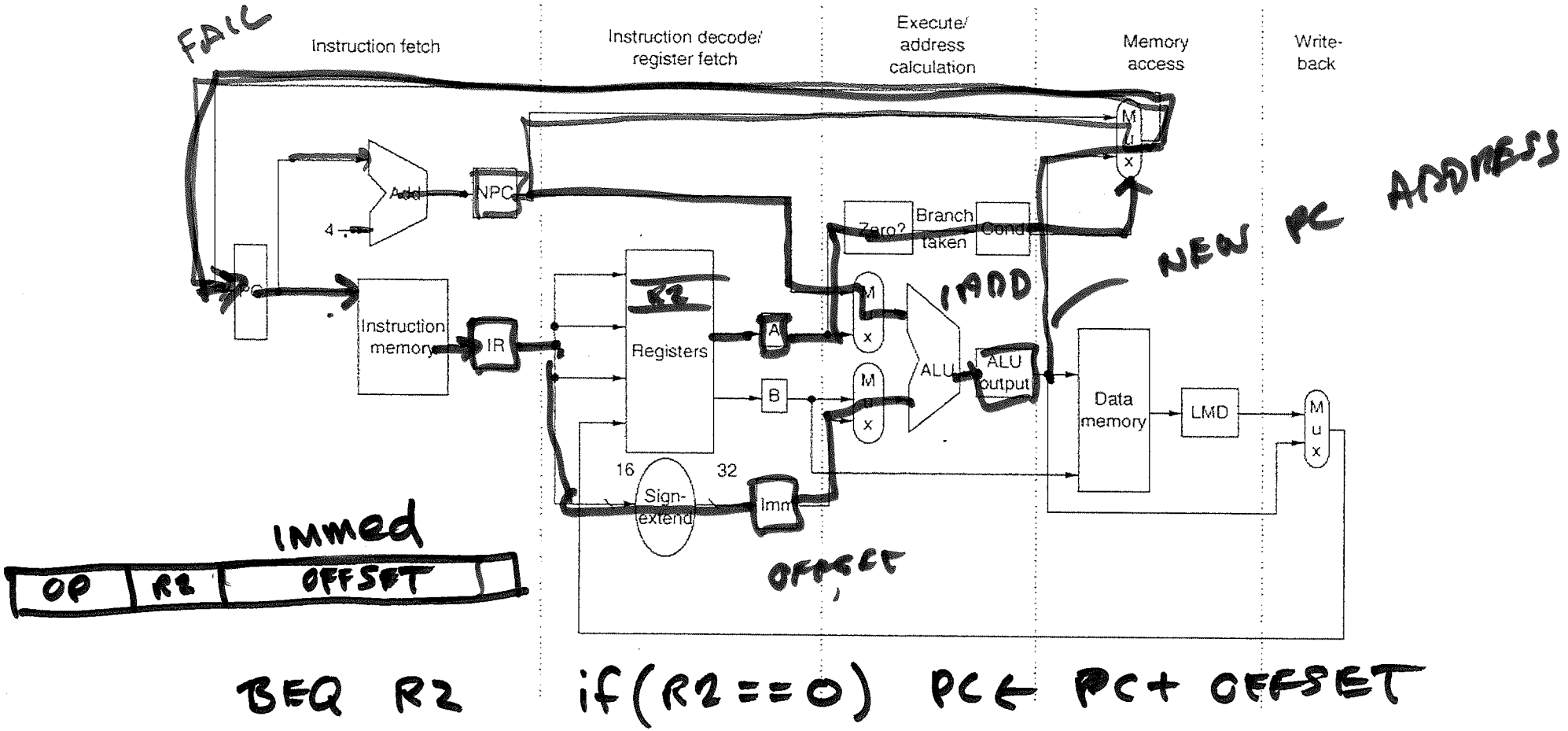
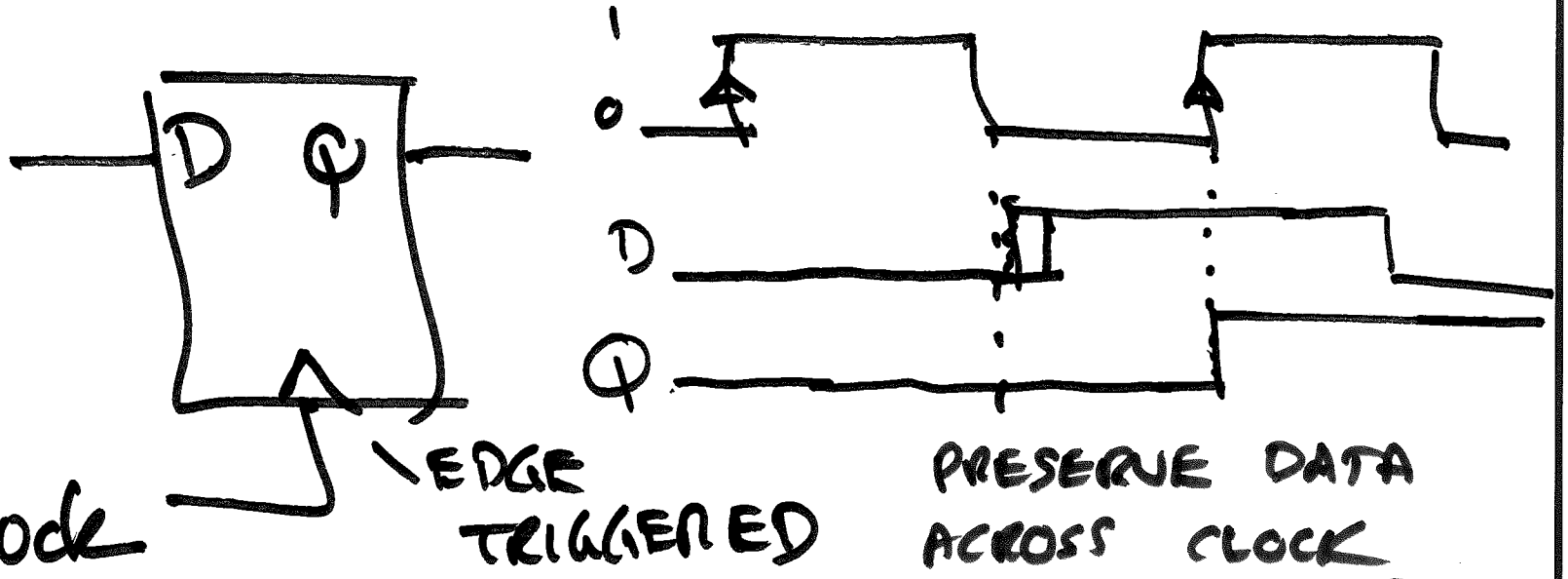


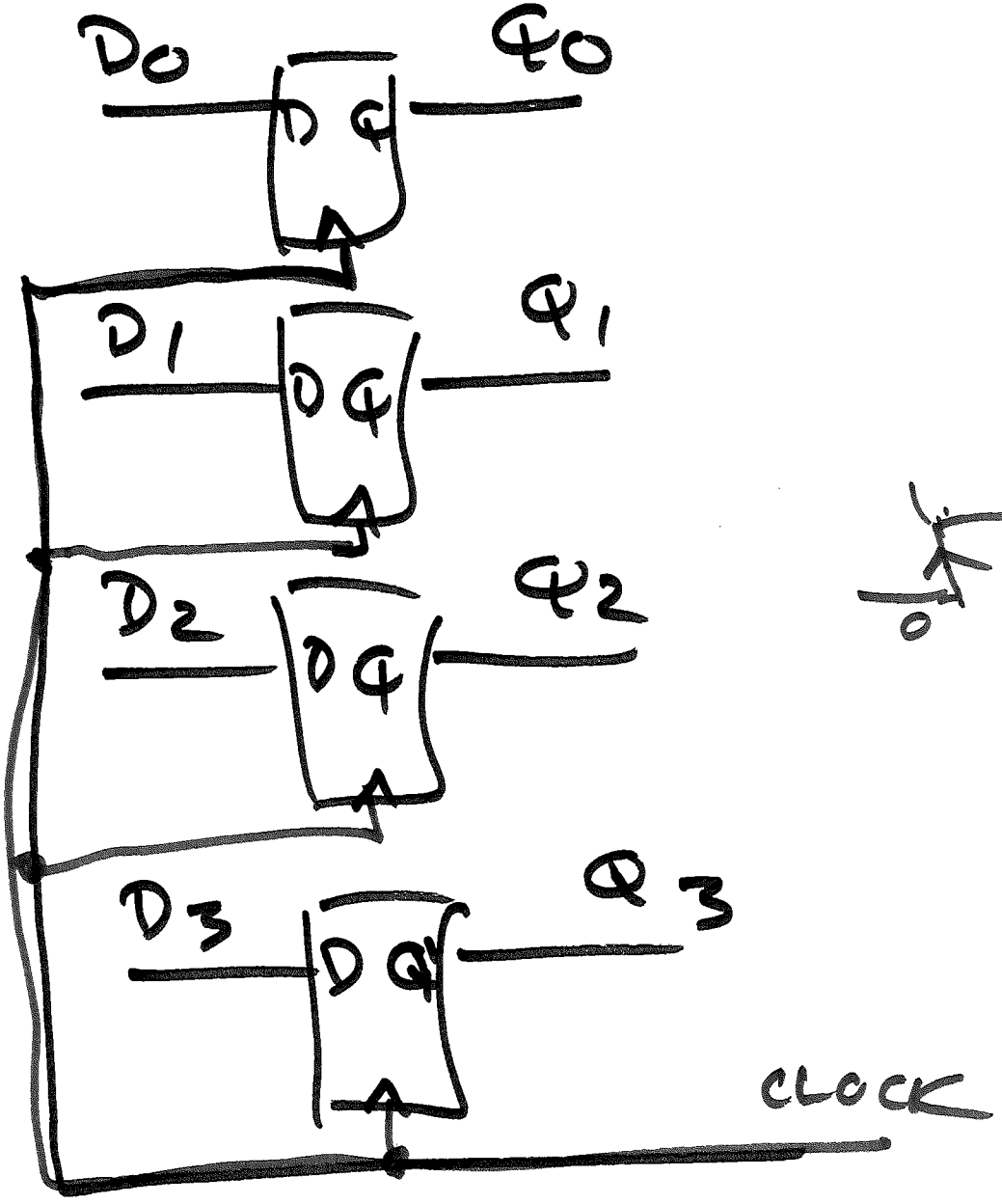
Figure C.21 The implementation of the MIPS data path allows every instruction to be executed in 4 or 5 clock cycles. Although the PC is shown in the portion of the data path that is used in instruction fetch and the registers are shown in the portion of the data path that is used in instruction decode/register fetch, both of these functional units are read as well as written by an instruction. Although we show these functional units in the cycle corresponding to where they are read, the PC is written during the memory access clock cycle and the registers are written during the write-back clock cycle. In both cases, the writes in later pipe stages are indicated by the multiplexer output (in memory access or write-back), which carries a value back to the PC or registers. These backward-flowing signals introduce much of the complexity of pipelining, since they indicate the possibility of hazards.

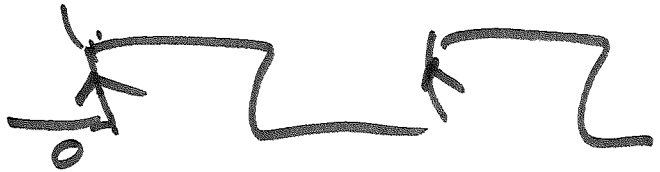
PIPELINED ~~INSTRUCTIONS~~ INSTRUCTIONS

PIPELINE REGISTERS - ISOLATE STAGES

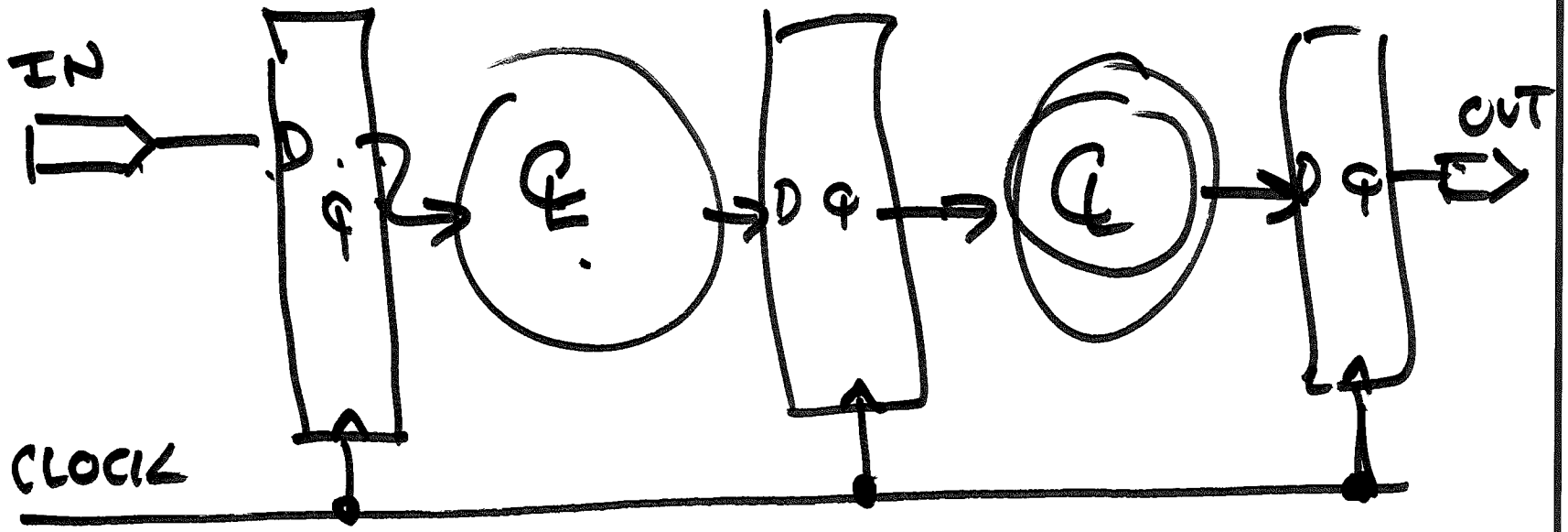
REGISTER: SET OF FLIP FLOPS w/ COMMON CLOCK

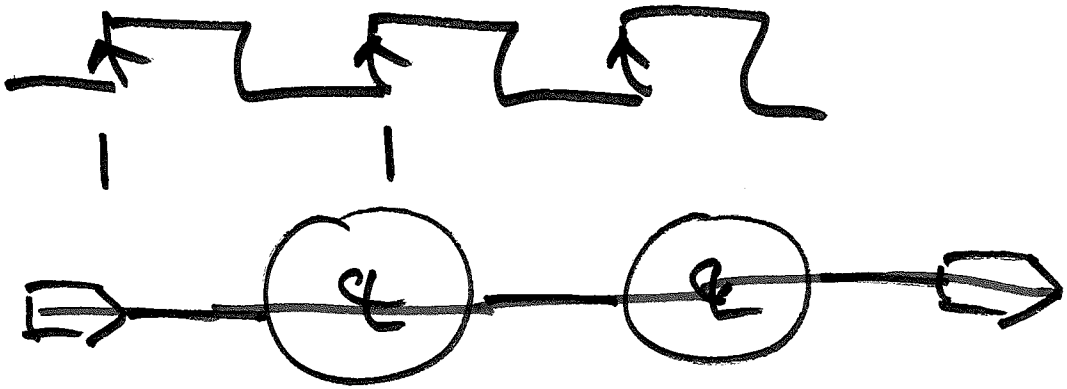



D0
Q0
D
Q
D1
Q1
D
Q
D2
Q2
D
Q
D3
Q3
D
Q
CLOCK



University of Idaho





University of Idaho

CHALLENGE

MAKE SURE WE DON'T NEED SAME RESOURCE FOR DIFFERENT OPERATIONS AT SAME TIME.

e.g. ALU – ADDR REGISTERS (DADD)

GENERATE ADDRESSES

- MEM –
- BR –

Q: Is this a problem for FIG. C.21 design, no pipelining

A: NO – DON'T NEED ALU AT THE SAME TIME FOR DIFFERENT INSTR.

PIPELINE HAZARDS C.2

1. STRUCTURE HAZARDS – H/W CAN'T SUPPORT ALL COMBINATIONS OF INSTR. AT SAME TIME – RESOURCE CONTENTION

2. DATA HAZARDS

```
DMUL  R1, R2    R1 ← R1 x R2
DADD  R1, R3    R1 ← R1 + R3
DADD depends on DMUL result
```

~~3. CONTROL HAZARDS~~

2

3. CONTROL HAZARDS

DSUB R1, R2

BEQ LOOP.

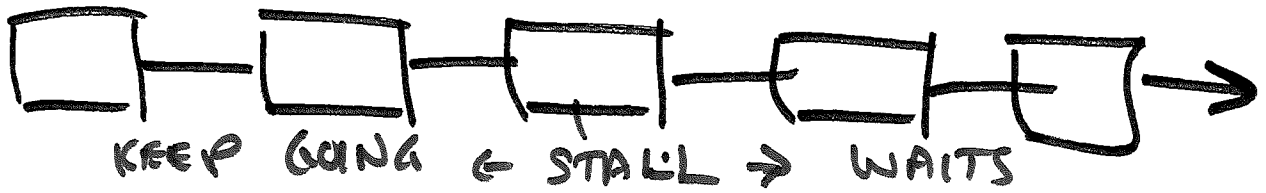
. Do we branch or not?

Don't know until DSUB completed.

WHAT DO WE DO WHEN WE HAVE A HAZARD?

STALL THE PIPELINE UNTIL HAZARD RESOLVED.

BUBBLE



HAZARDS INTRODUCE PIPELINE "BUBBLES"

- REGIONS OF NO WORK
- FLOW THRU PIPELINE

- REDUCE PERFORMANCE ADVANTAGE OF PIPELINING

SOLUTIONS

1. HARDWARE
2. SOFTWARE
   - INSTRUCTION RE-ORDERING