CS 451 / 551 / ECE 541

ADVANCED
COMPUTER ARCHITECTURE

SESSION no. 8

University of Idaho

PIPELINE HAZARDS

- STRUCTURAL — RESOURCES
- DATA - DEPENDENCY
- CONTROL - CONDITIONAL BRANCHES

STALL PIPELINE - WAIT
REDUCE SPEED ADVANTAGE
OR
HARDWARE / SOFTWARE SOLUTIONS TO AVOID STALLING

EX C-14. DATA REFS 40% OF INSTR.
CPI = 1 NO HAZARDS
ASSUME: PROCESSOR W/HAZARDS
CLOCK 1.05 × NO. HAZARDS CLOCK

University of Idaho

REDUCE HAZARDS ⇒ EXTRA CIRCUITS

⇒ LONGER DELAY ⇒ SLOWER CLOCK.

WHAT IS PAYOFF OF EXTRA EFFORT TO REDUCE HAZARDS

AVG. INSTR. TIME = CPI × CLOCK CYCLE TIME

P1: NO STALLS, NO HAZARDS

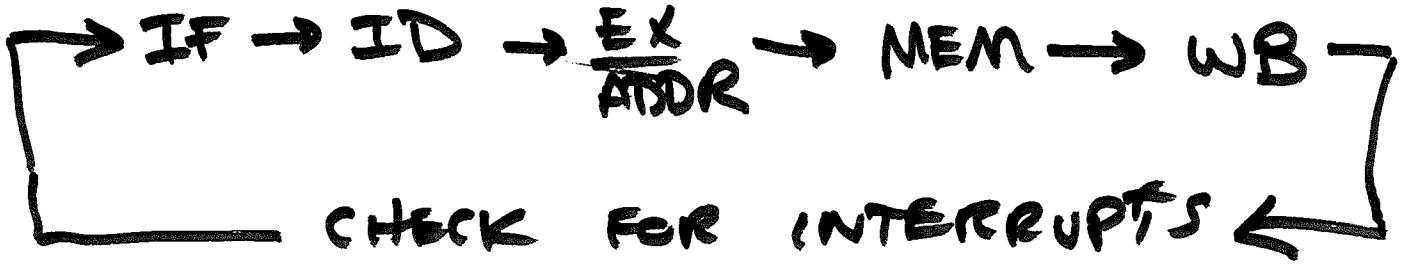
% DATA INSTR

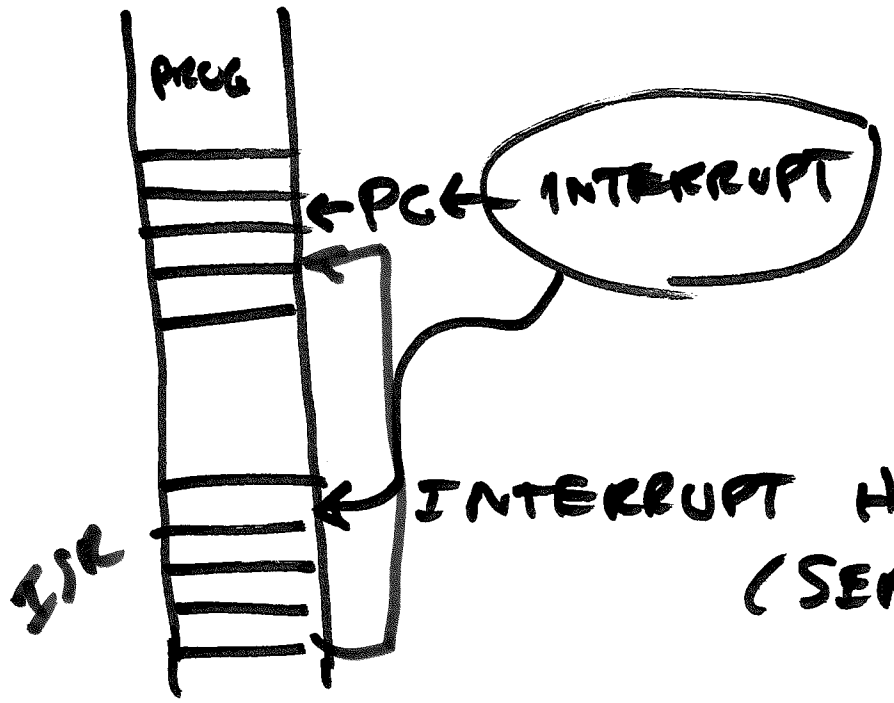
AVG INTR. TIME = $(1 + 0.4 \times 1)$ =

$$\frac{\text{CLOCK CYCLE TIME}}{1.05} = 1.3$$

NO HAZARDS: 1.3× SLOWER THAN HAZARDS.

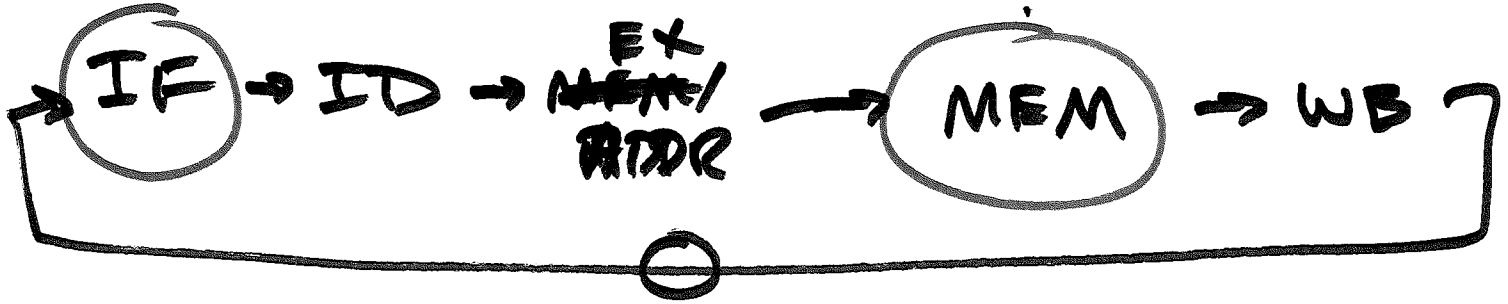
PIPELINING HAS TO WORK WITH INTERRUPTS





- CHANGE FLOW OF EXECUTION
  - ASYNCHRONOUSLY
- CHANGE CONTEXT
  - SAVE REG
  - RAISE PRIVELEDGE

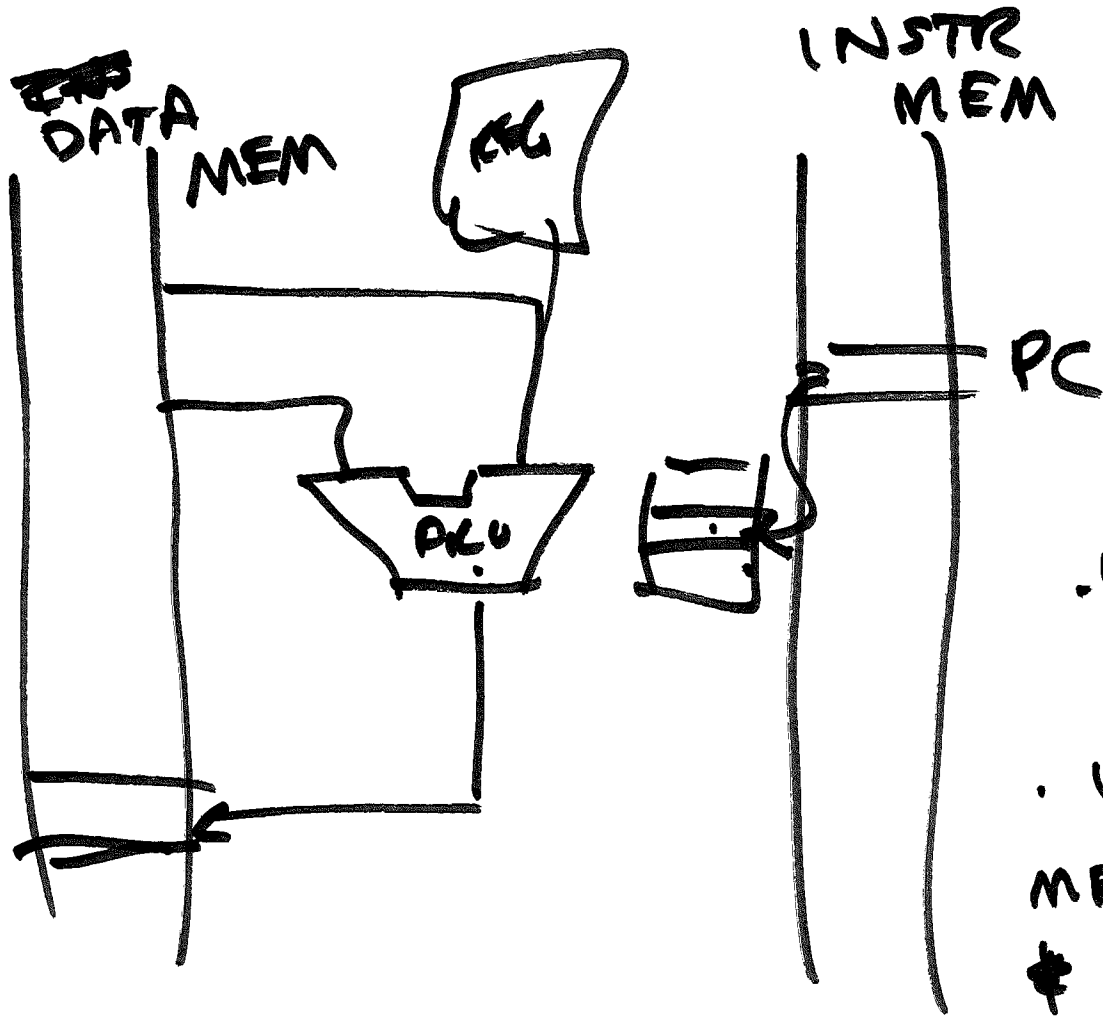
EXAMPLE:



IF & MEM ACCESS MEMORY IN SAME CLOCK CYCLE

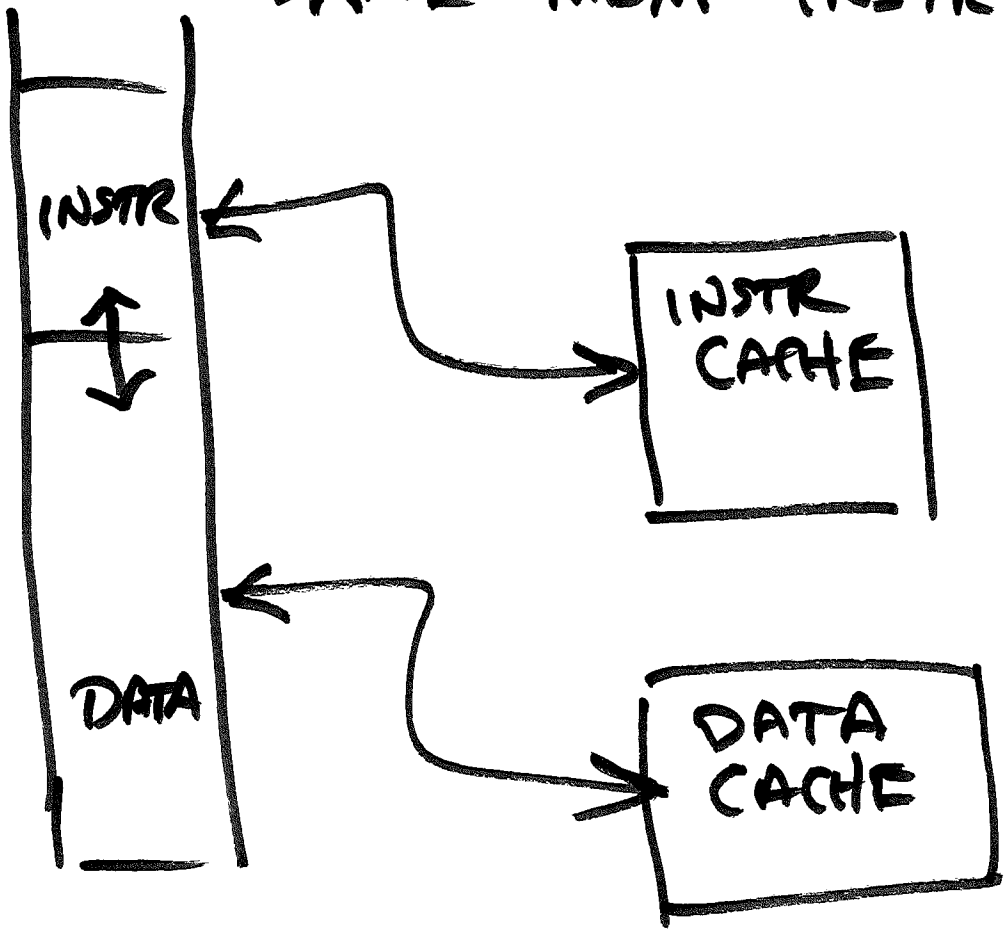
- MEMORY ACCESS CONFLICT
  1) STALL
  2) SEPARATE "I & D" ~~SPACE~~ MEMORY
     HARVARD ~~ARCHITECTURE~~ MEMORY

Q: LIMITATION W/ ~~HARVARD~~ ARCH?



- WASTED MEMORY!
- USE SAME MEM FOR INSTR & DATA - CAN USE ALL OF IT!

SAME MEM INSTR & DATA.



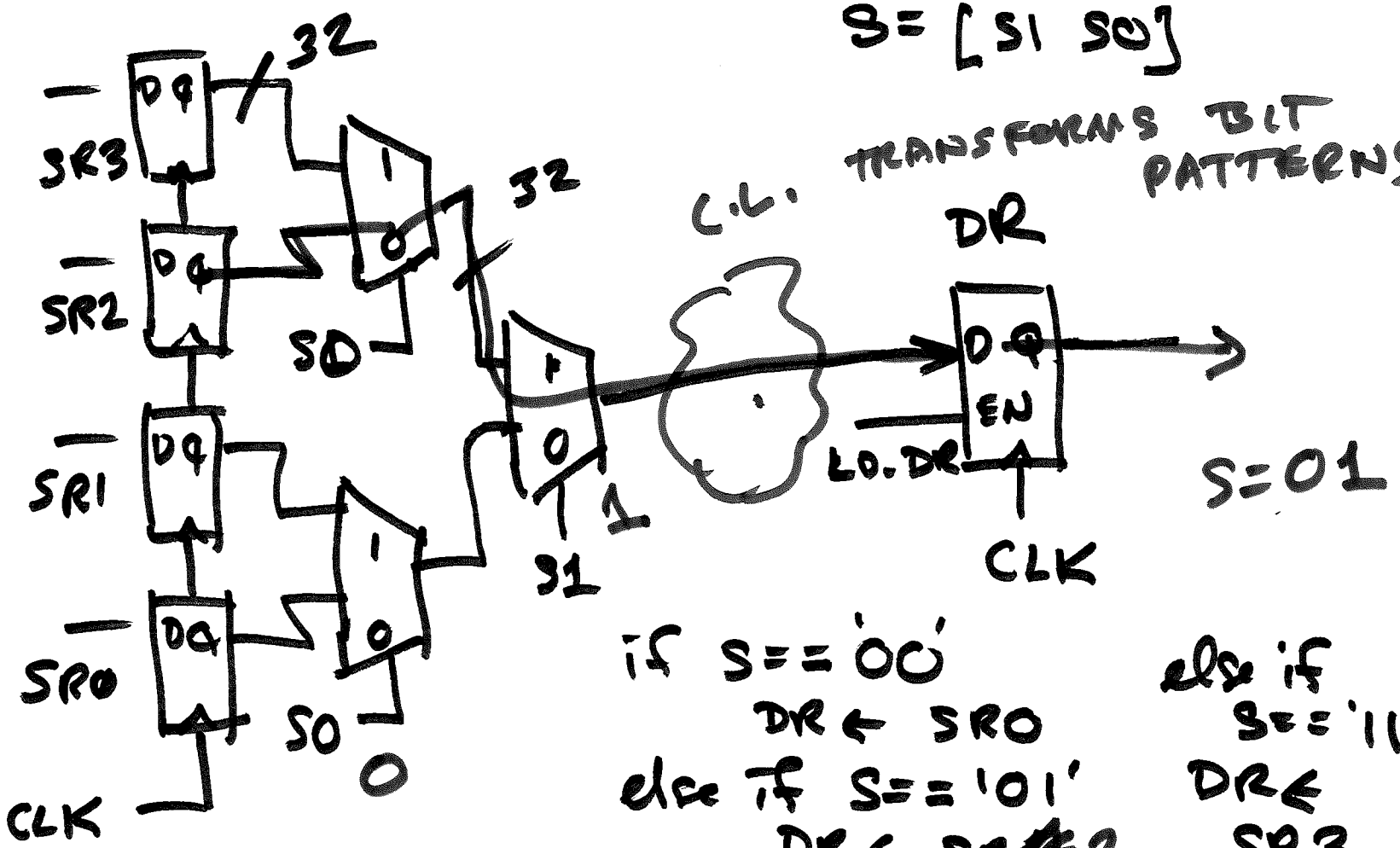
STRUCTURAL: EXTRA RESOURCES

(DUPLICATE HARDWARE)

MEM, LOGIC BLOCKS, REGISTERS...

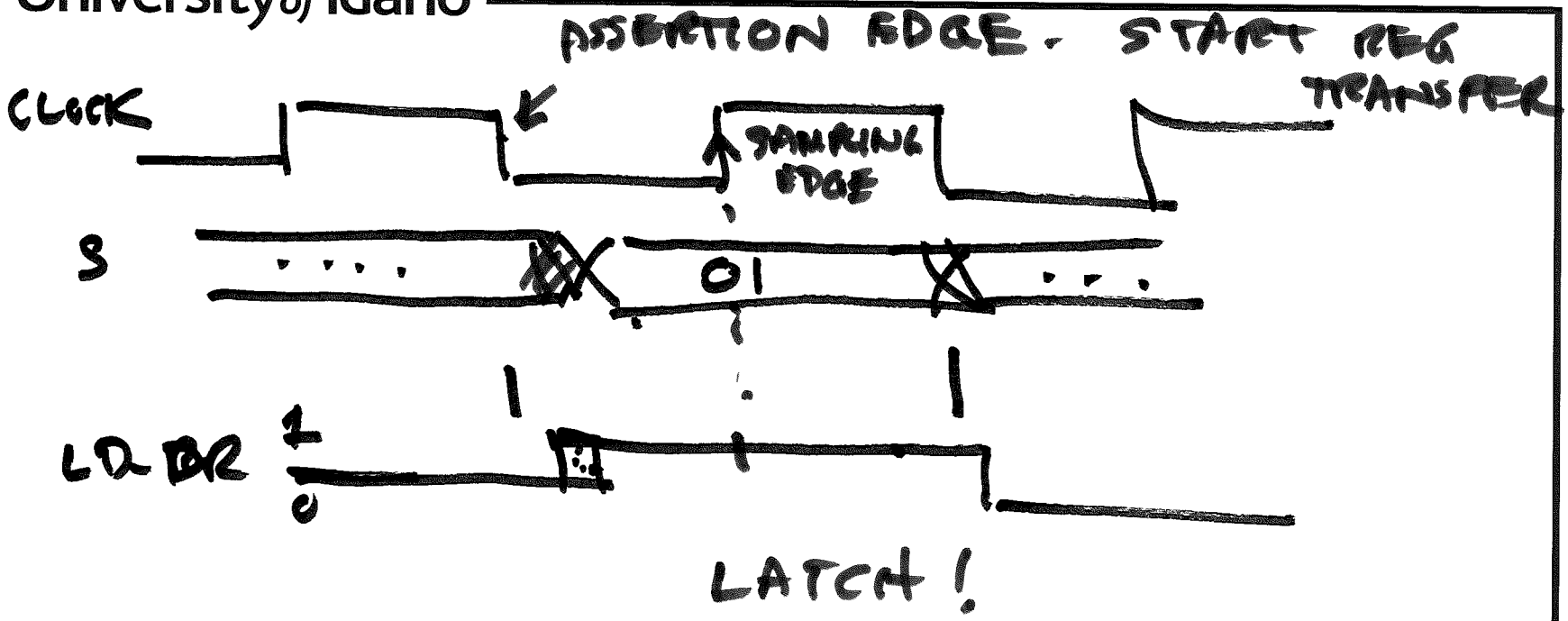
- EXTRA INTERCONNECT — "ROUTABILITY"

1. MULTIPLEXERS

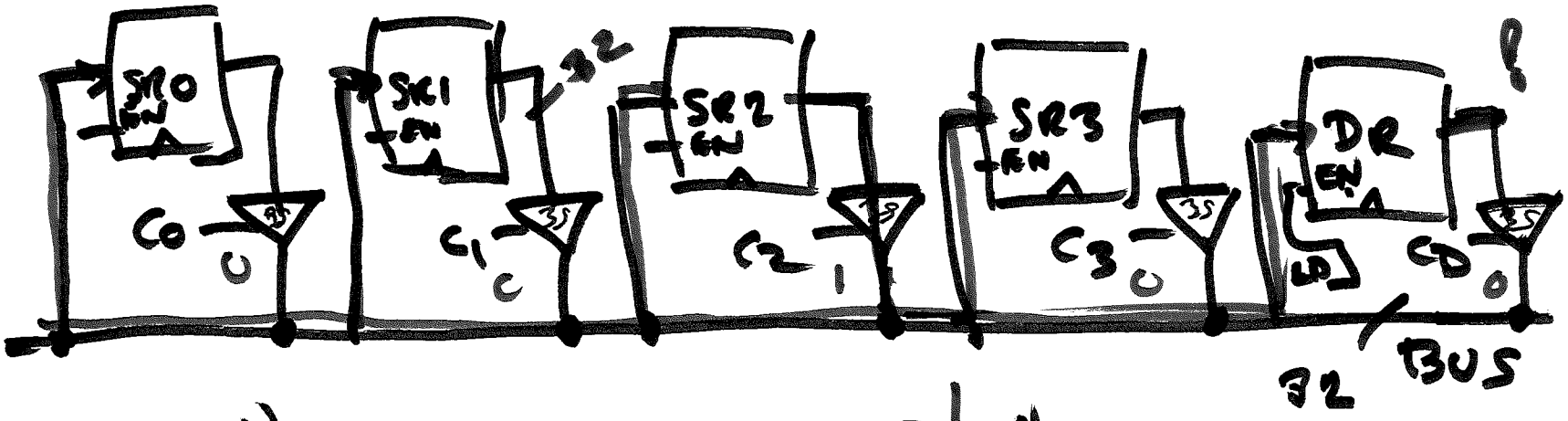


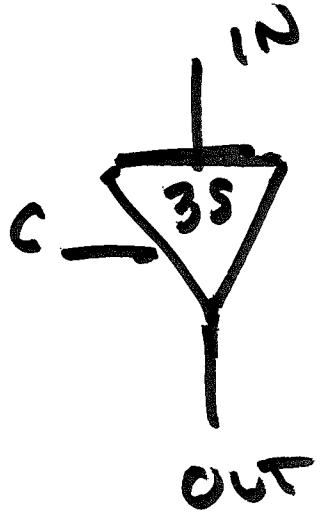

```
if S == '00'
    DR ← SR0
else if S == '01'
    DR ← SR >> 2
else if S == '10'
    DR ← SR >> 1
else if S == '11'
    DR ← SR3
```

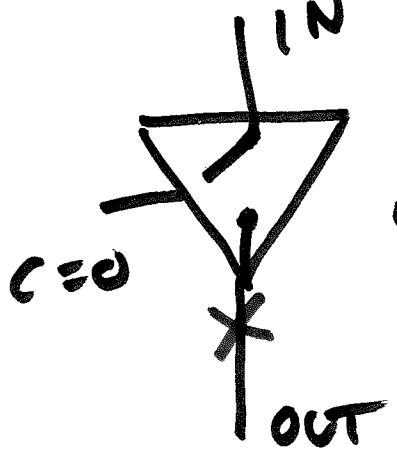
University of Idaho

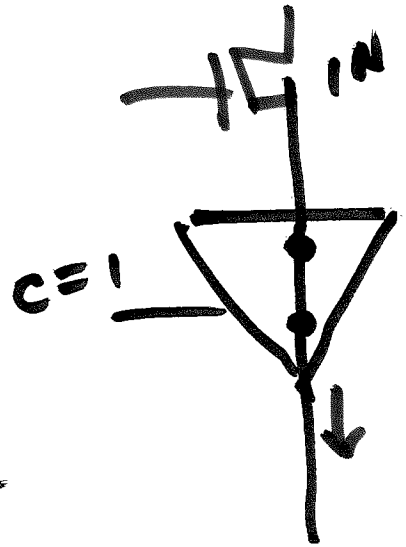


2. SHARED BUS





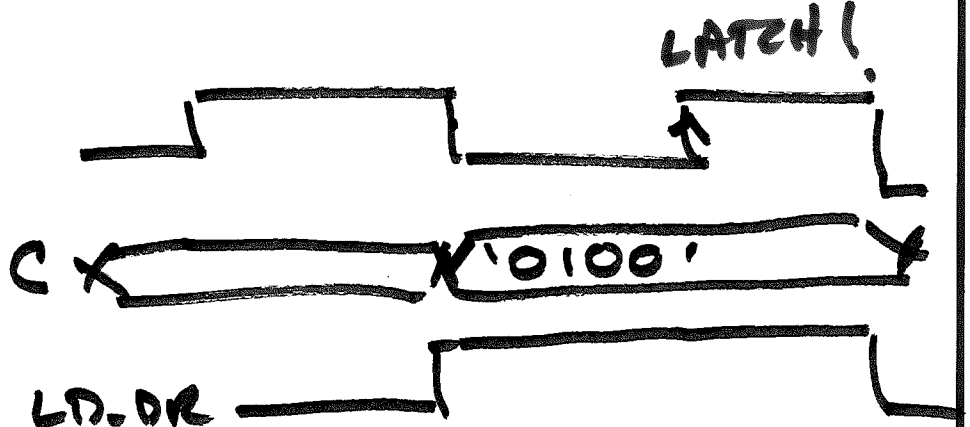




ELECTRICALLY
~~PHYSICALLY~~
DISCONNECT
INPUT FROM
OUTPUT

DR ← SR2

```
if C0 == '1'                ; AND C1, C2, C3, CD = 0
    R0 ← SR0
if C1 == '1'
    DR ← SR1
        :
if C3 == '1'
    DR ← SR3
```



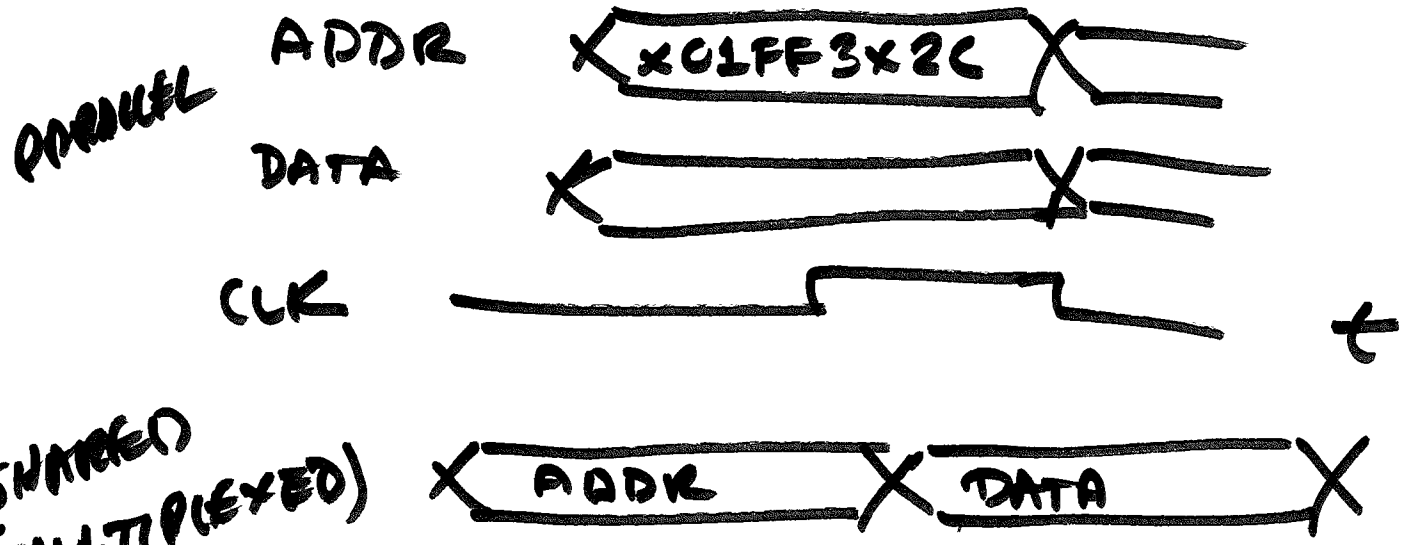
SHARED BUS: ADVANTAGE(S)

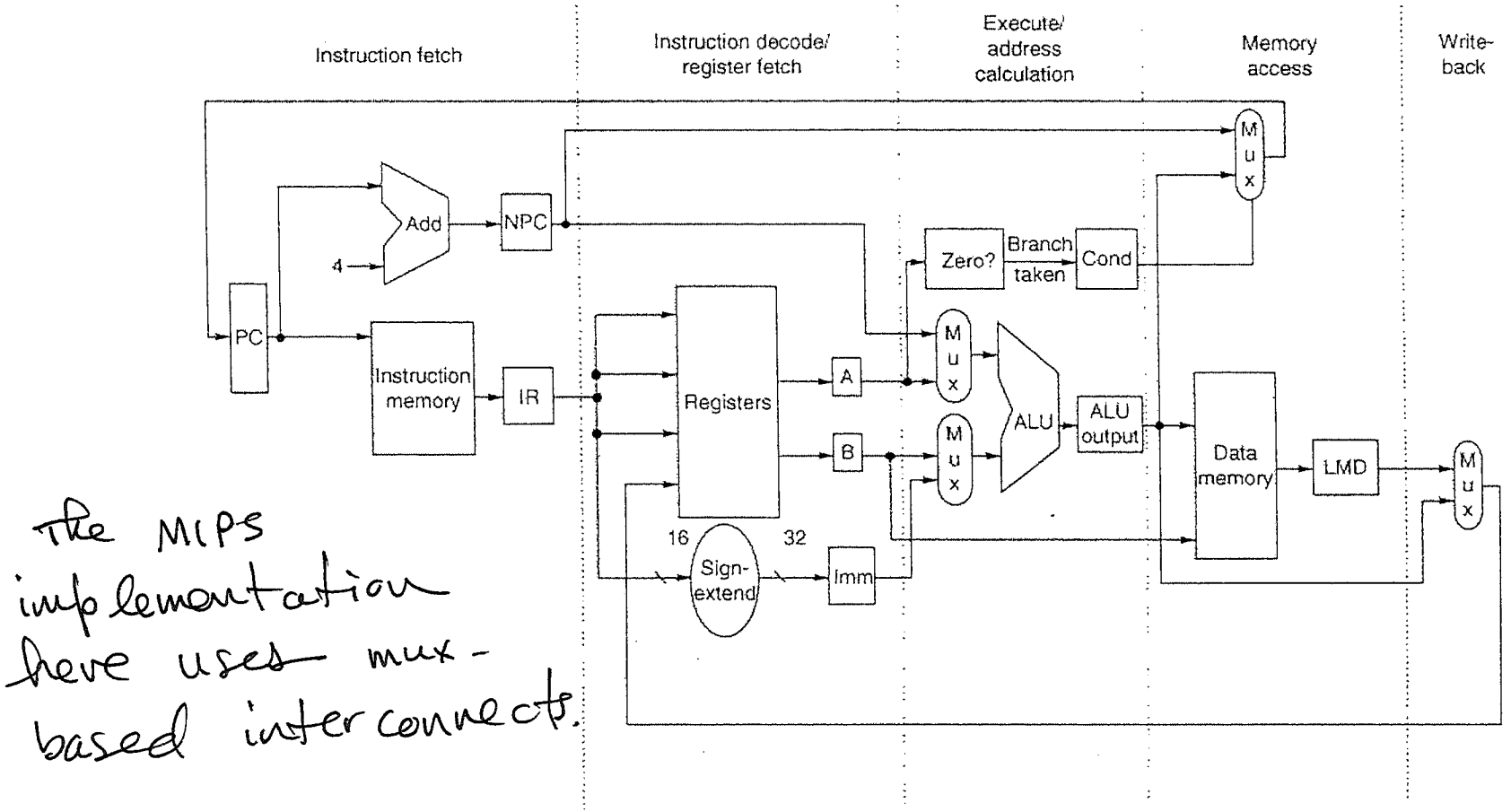
LESS WIRING - SHARE ROUTING

S. B. . DISADVANTAGE(S)

SLOWER - ONLY ONE DEVICE CAN "TALK" AT A TIME.

EX ~~MEMORY~~ MEMORY ACCESS





The MIPS implementation here uses mux-based interconnects.

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.

C-16 DATA ~~HAZ~~ HAZARDS

DATA DEPENDENCY

DADD R1, R2

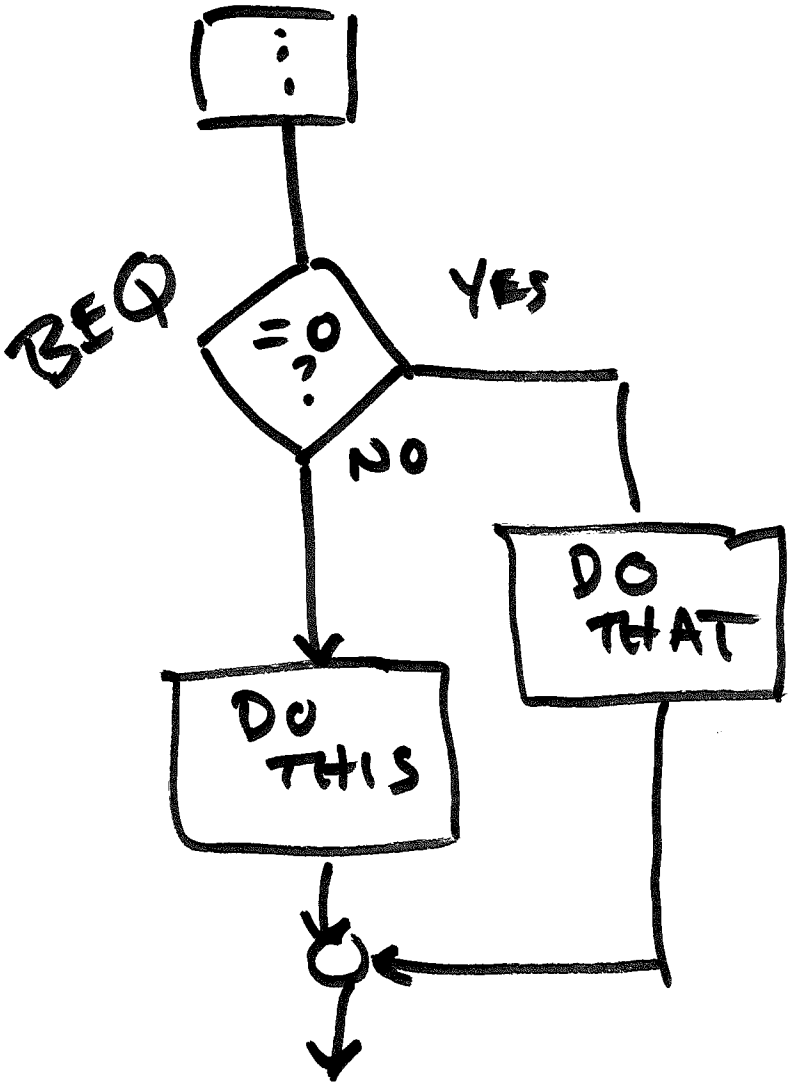
DSUB R4, R3

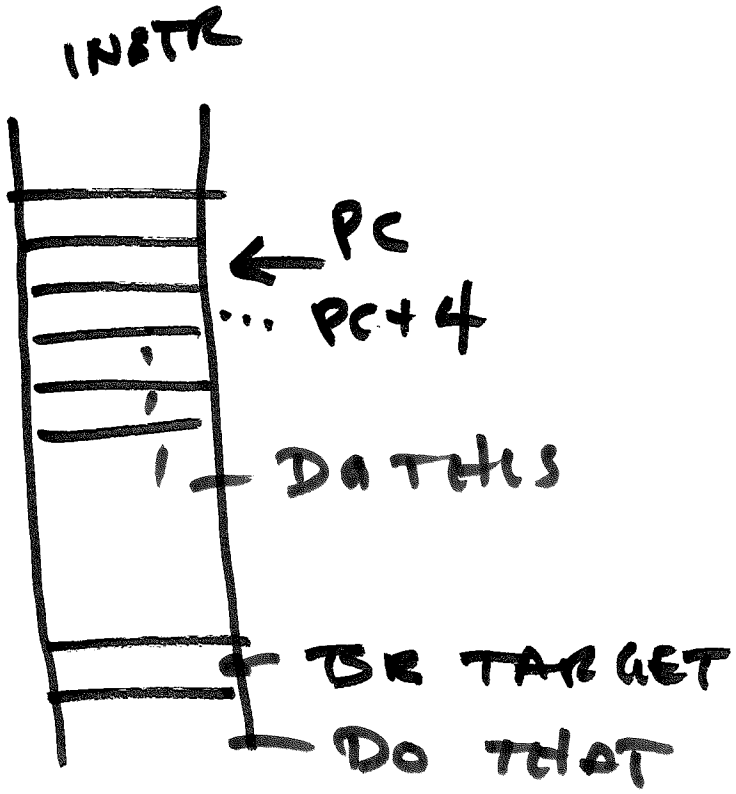
FIXES?

- INSTRUCTION REORDERING
- FORWARDING
  - MOVE THE ADD RESULT IN THE PIPELINE TO THE STAGE THAT NEEDS IT.
  - CHANGE IN HDWE DESIGN.
  - BEGINNING OF "OUT OF ORDER" EXECUTION.

University of Idaho

BRANCH HAZARDS





CHOICE:

PC ← PC+4
OR
PC ← BRANCH TARGET

STRATEGIES

1. FREEZE OR FLUSH - DON'T FILL PIPELINE UNTIL WE ~~KOW~~ KNOW: 'TAKEN' OR NOT

2. PREDICT: TAKEN
   FETCH TARGET INSTR.

3. PREDICT: NOT TAKEN
   FETCH PC + 4

4. REORDERING - FILL PIPELINE WITH INSTRUCTIONS THAT HAVE TO BE EXECUTED ANY WAY.