how long will it take to make $N$ widgets?

If we pipeline the process into $S$ stages,

Given what we know about pipelining, assume in

Review Problem 28
Divide datapath into multiple pipeline stages

Pipelined Datapath
Control signals for WR (RegWE, Mem2Reg, MemWE) are used 3 cycles later.
Control signals for Mem (MemWE, Mem2Reg) are used 2 cycles later.
Control signals for Exe (ALUop, ALUSRC) are used 1 cycle later.

The Main Control generates the control signals during Reg/Dec.

Pipelined Control
take action (or delay action) to resolve hazards
pipeline control must detect the hazard

Can always resolve hazards by waiting
branch instructions

need to see after dryer before next load in

E.G. washing football uniforms and need to get proper detergent level.
control hazards: attempt to make decision before condition evaluated

institution depends on result of prior instruction still in the pipeline

.sock from washer through dryer

E.G. one sock of pair in dryer and one in washer; can't fold until get
data hazards: attempt to use item before it is ready

E.G. combined washer/dryer would be a structural hazard or folder

at the same time
structual hazards: attempt to use the same resource two different ways

yes: Pipeline Hazards

Can pipelining get us into trouble?
Register File's Write port (bus W) for the WR stage
Data Memory for the Mem stage
ALU for the Exec stage
Register File's Read ports (bus A and busB) for the Reg/Dec stage
Instruction Memory for the Fetch stage
Instruction Memory in the pipeline datapath are:

Pipelining the Load Instruction
The Four Stages of R-Type

- Fetch: Fetch the instruction from the instruction memory.
- Dec: Register Fetch and Instruction Decode.
- Exec: ALU operates on the two register operands.
- WR: Write the ALU output back to the register file.
Interaction between R-type and loads causes structural hazard on writeback.
Mem stage is a **NOOP** stage: nothing is being done.

Now R-type instructions also use `Reg File`s write port at Stage 5.

Solution: Delay R-type`s register write by one cycle.

R-type uses Register File`s write port during its 4th stage.

Load uses Register File`s write port during its 5th stage.

Each functional unit must be used at the same stage for all instructions:

Each functional unit can only be used once per instruction.

**Important Observation**
Pipelining the R-Type Instruction
The Four Stages of Store

- Fetch: Fetch the instruction from the instruction memory
- Decode: Register, fetch and instruction decode
- Execute: Calculate the memory address
- Mem: Write the data into the data memory
- WR: Noop

Compatible with Load & R-type instructions
The Stages of Conditional Branch

- Fetch: Fetch the instruction from Memory
- Decode: Register Fetch and Instruction Decode, compute branch target
- Exec: Test condition & update the PC
- Mem: NOOP
- WR: NOOP

Beg | Ietch | Reg/Dec | Exec | Mem | WR
Branch updates the PC at the end of the Exec stage.

Control Hazard
Accelerate Branches

When can we compute the CBO condition? When can we compute branch target address?
Branch updates the PC at the end of the Reg/Dec stage.

Control Hazard 2
\[ CPI = 80\% \times 1 + 20\% (2) = 0.8 + 0.4 = 1.2 \]

If all other instructions take 1 cycle, and branches are 20% of instructions, CPI: 

Solution #1: Stall
CPI: 50% of branches actually not taken, and branch frequency 20%?

Solution #2: Branch Prediction