# Pipelining

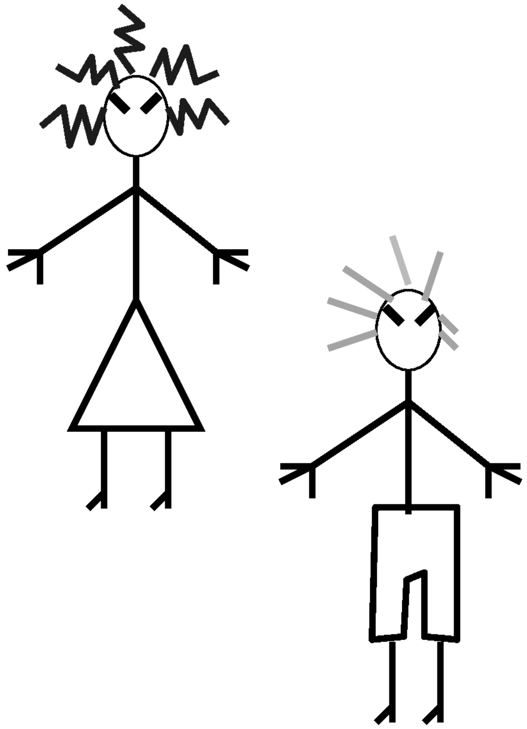
**Kevin Walsh**
**CS 3410, Spring 2010**
Computer Science
Cornell University

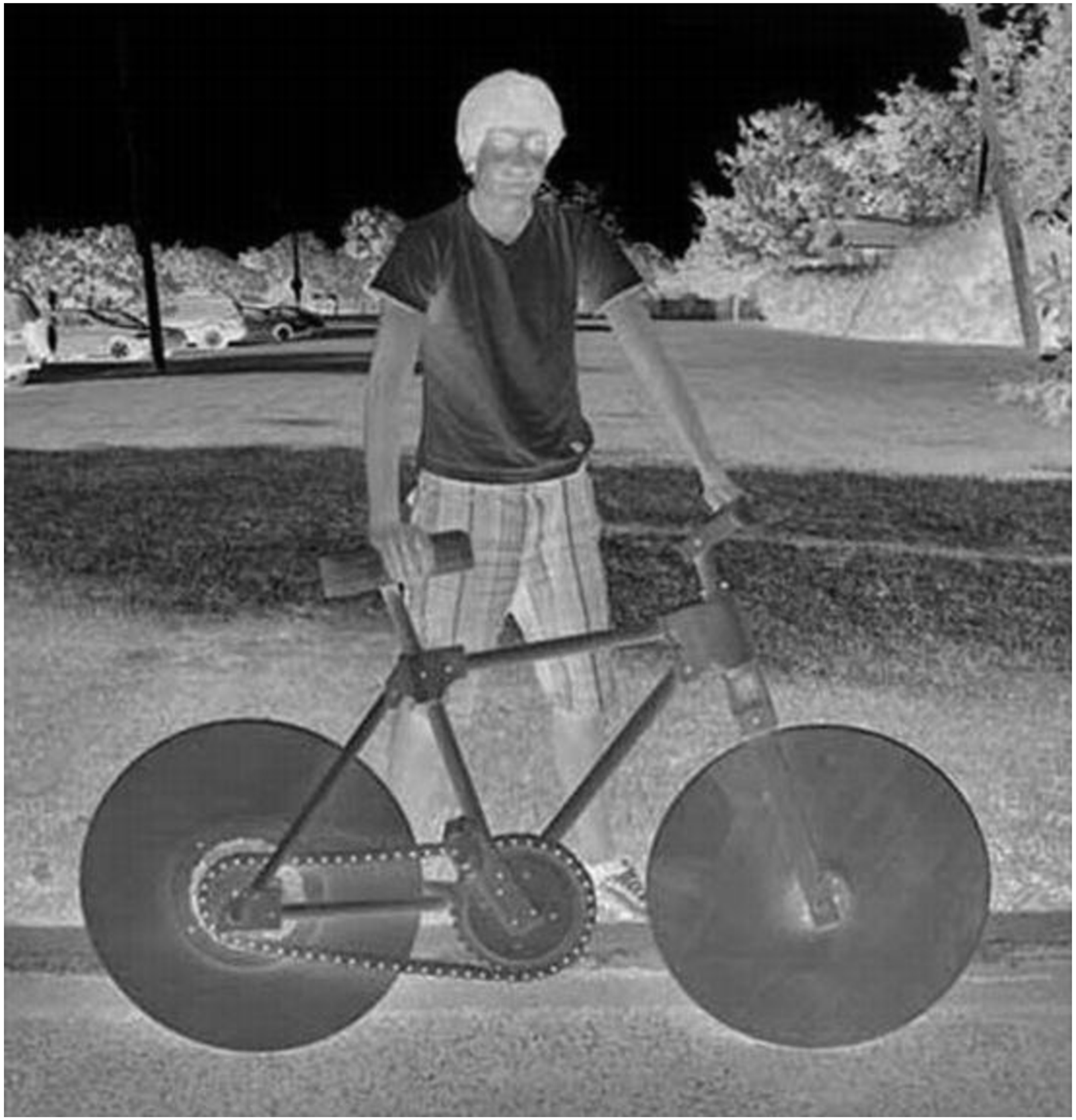
See: P&H Chapter 4.5

Alice

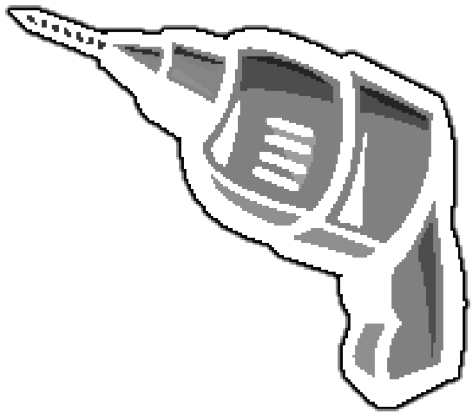
Bob

They don't always get along…





Saw

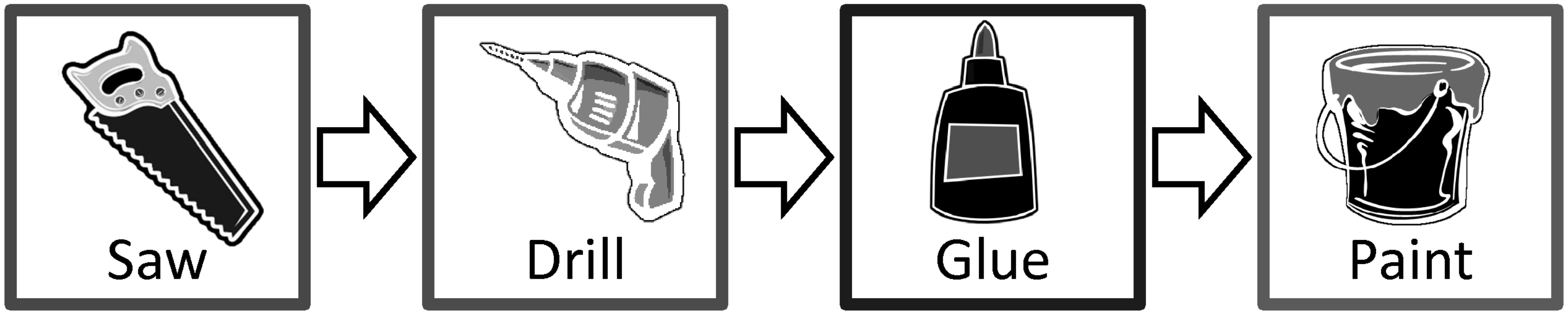


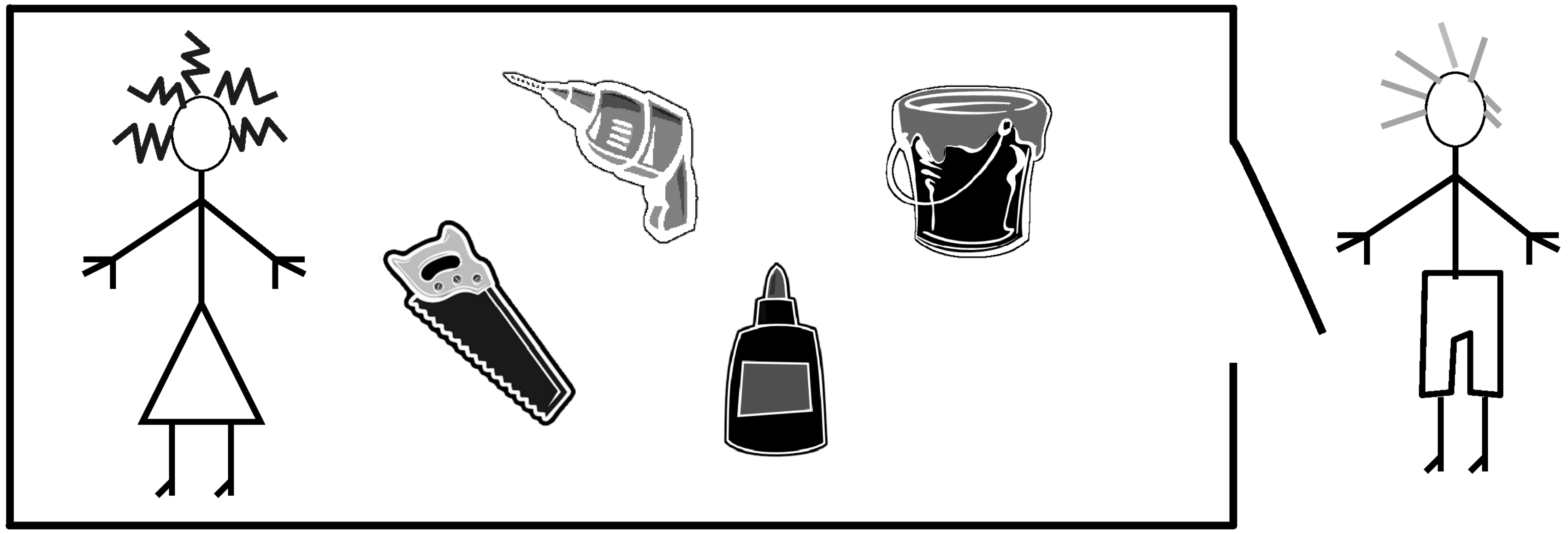
Drill

Glue

Paint

## N pieces, each built following same sequence:



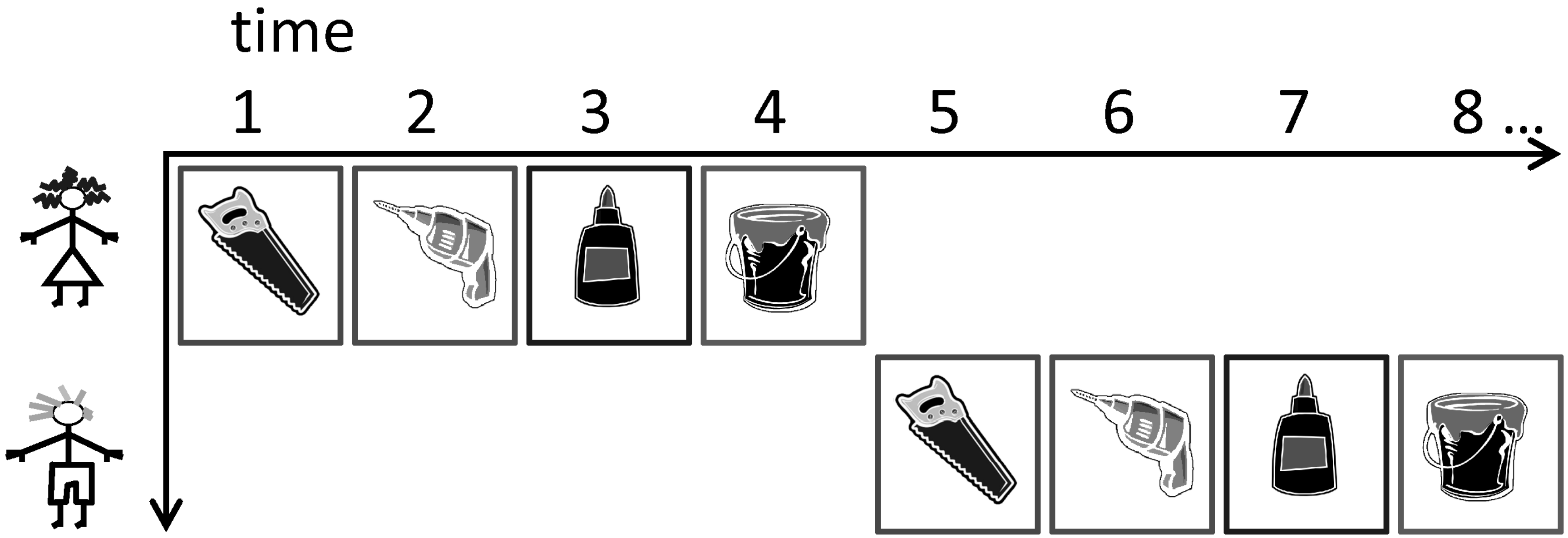


Alice owns the room

Bob can enter when Alice is finished

Repeat for remaining tasks

No possibility for conflicts



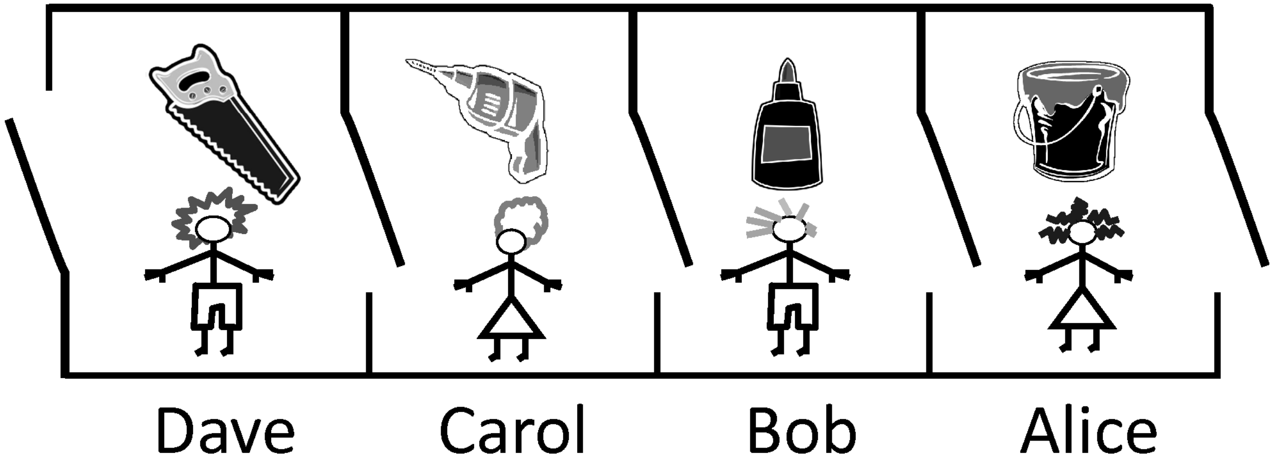
Latency:

Throughput:

Concurrency:

Can we do better?

## Partition room into *stages* of a *pipeline*

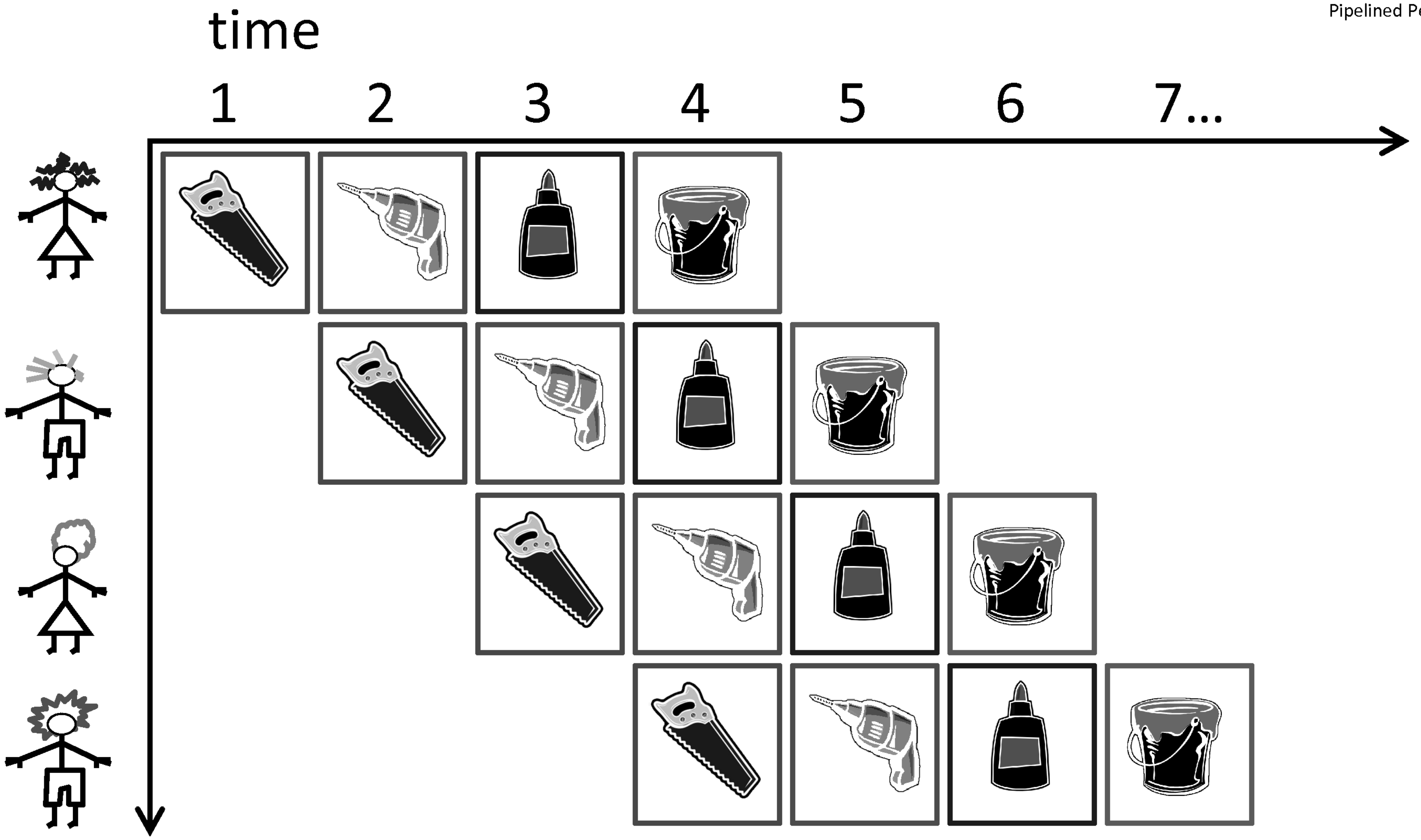


One person owns a stage at a time

4 stages

4 people working simultaneously

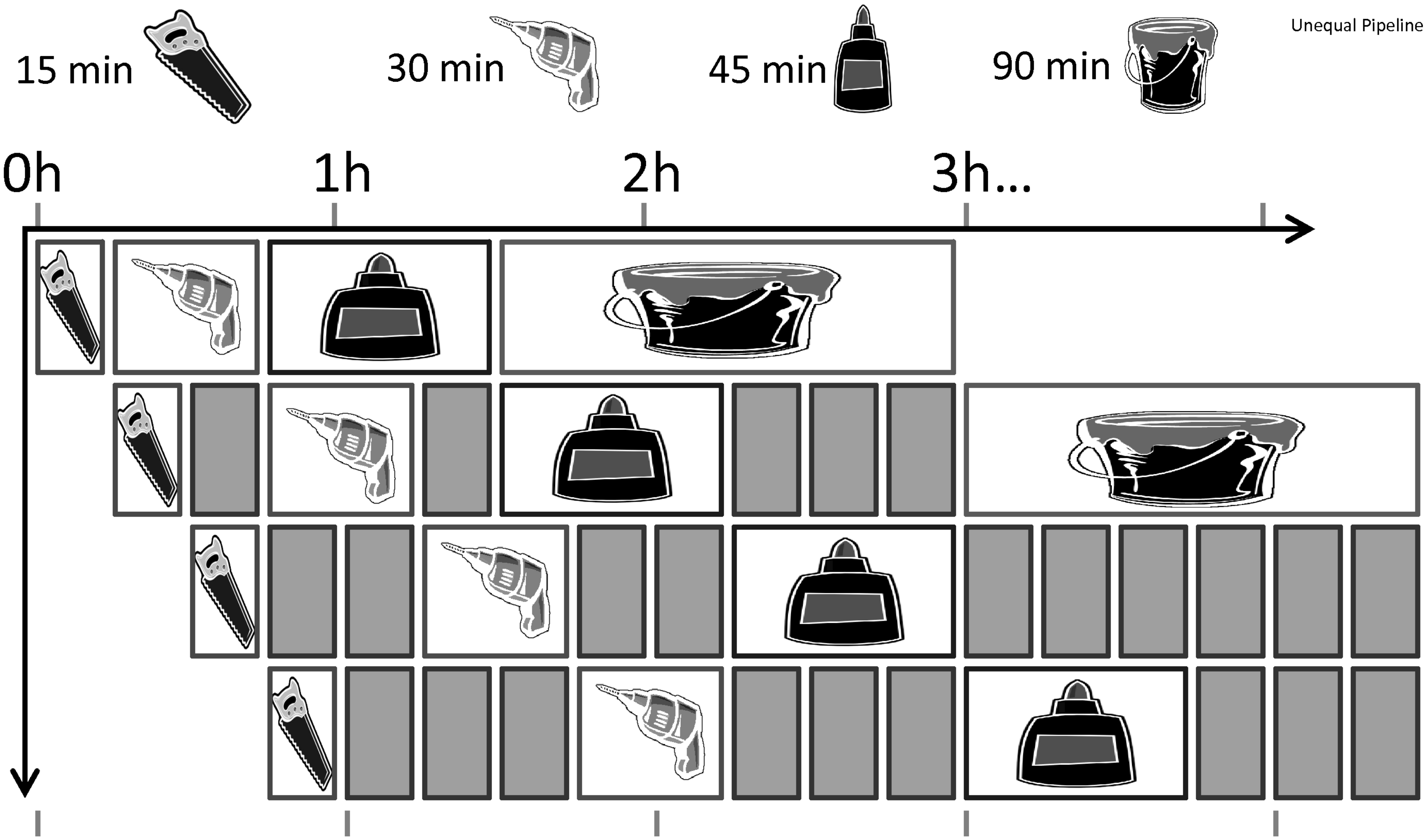
Everyone moves right in lockstep



Latency:

Throughput:

Concurrency:



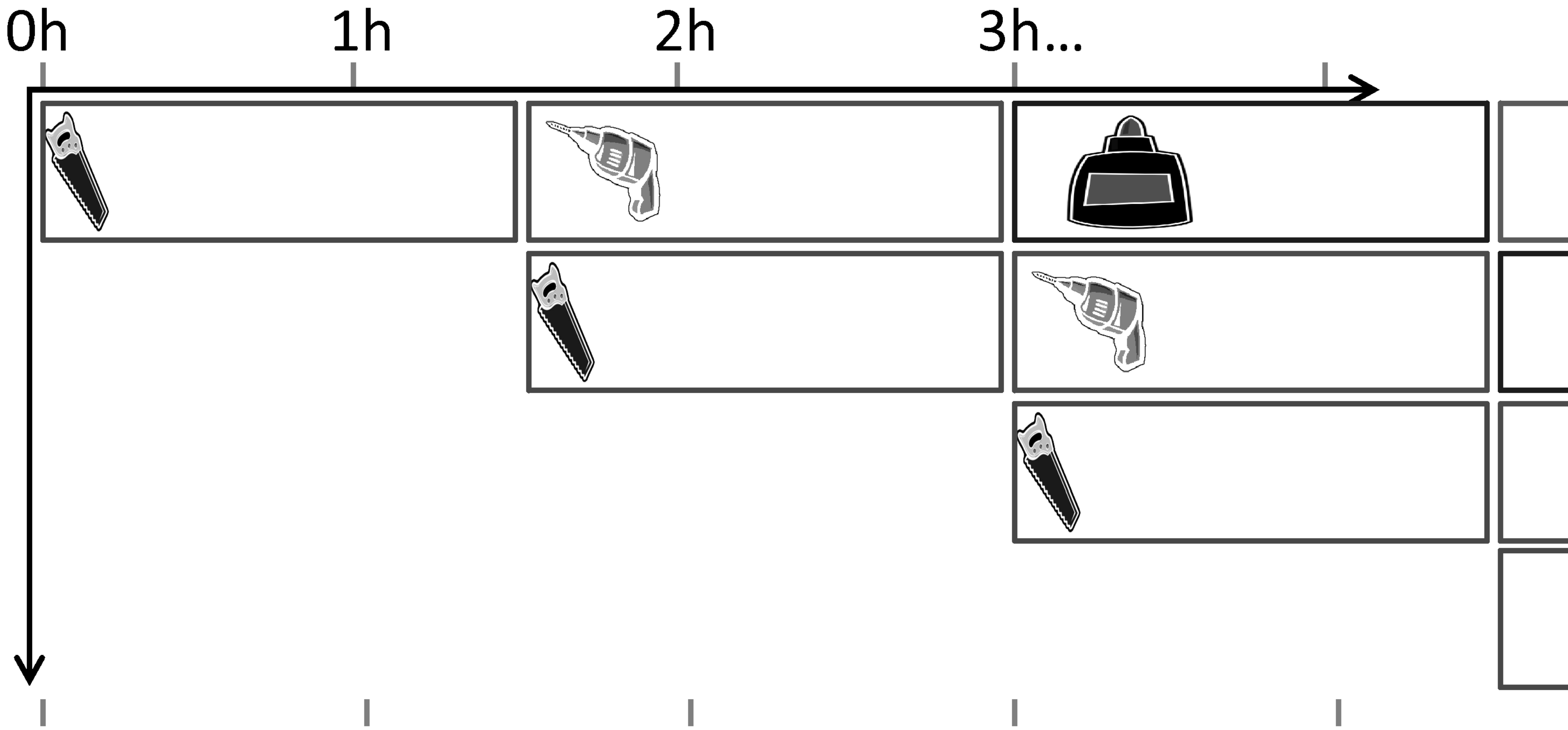
Latency:

Throughput:

Concurrency:

Poorly-balanced Pipeline Stages

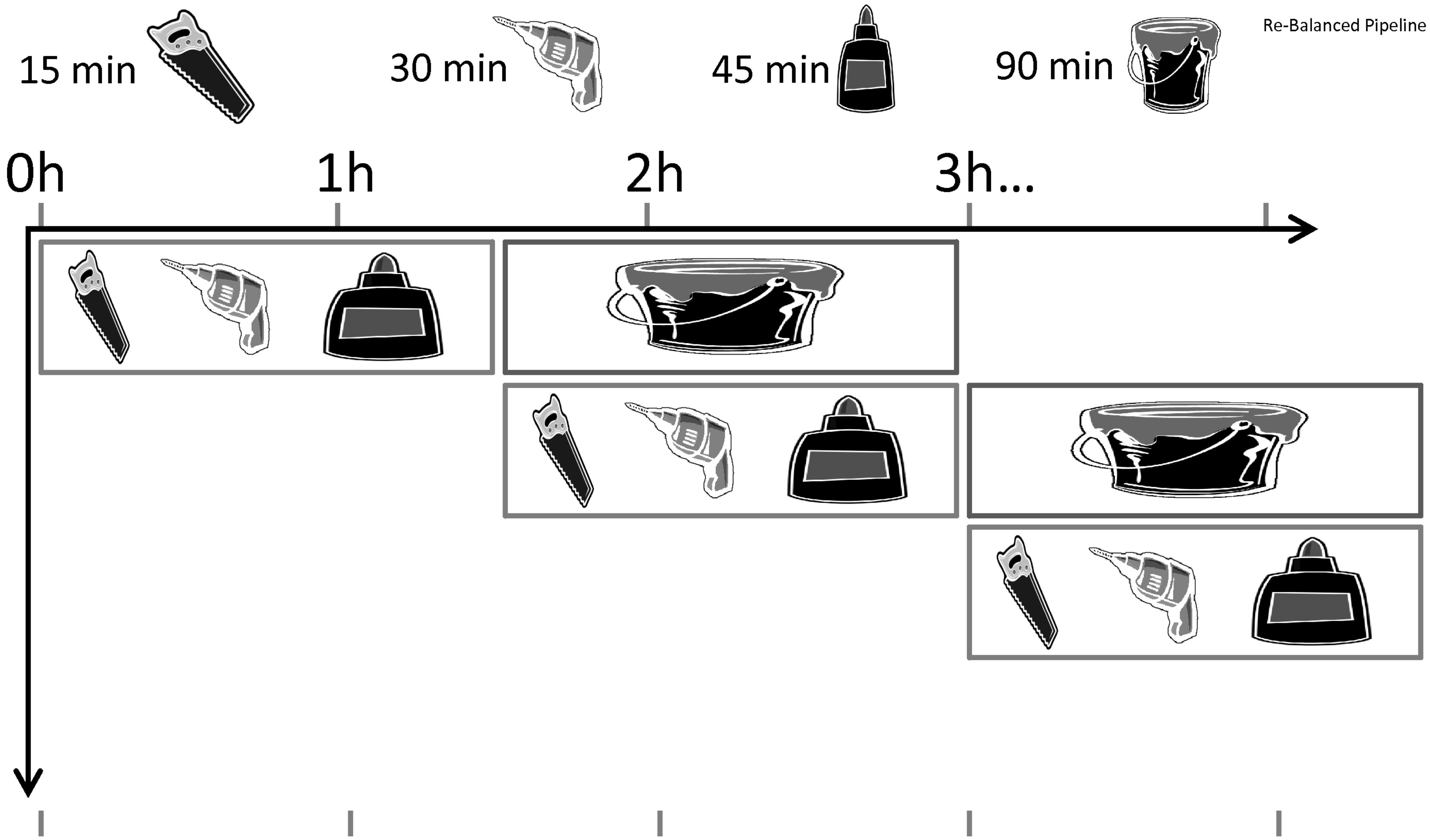
15 min
30 min
45 min
90 min



Latency:

Throughput:

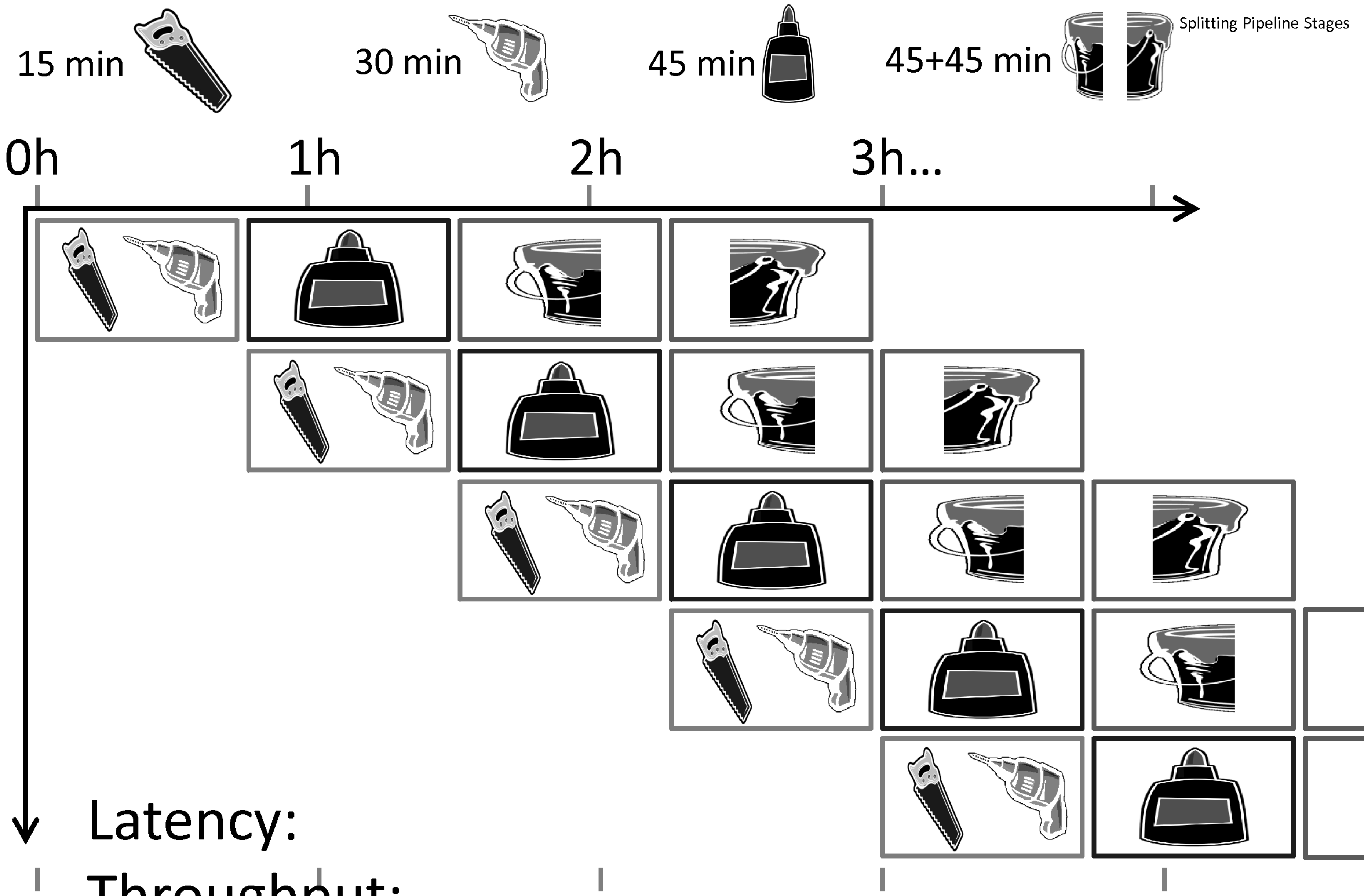
Concurrency:



Latency:

Throughput:

Concurrency:

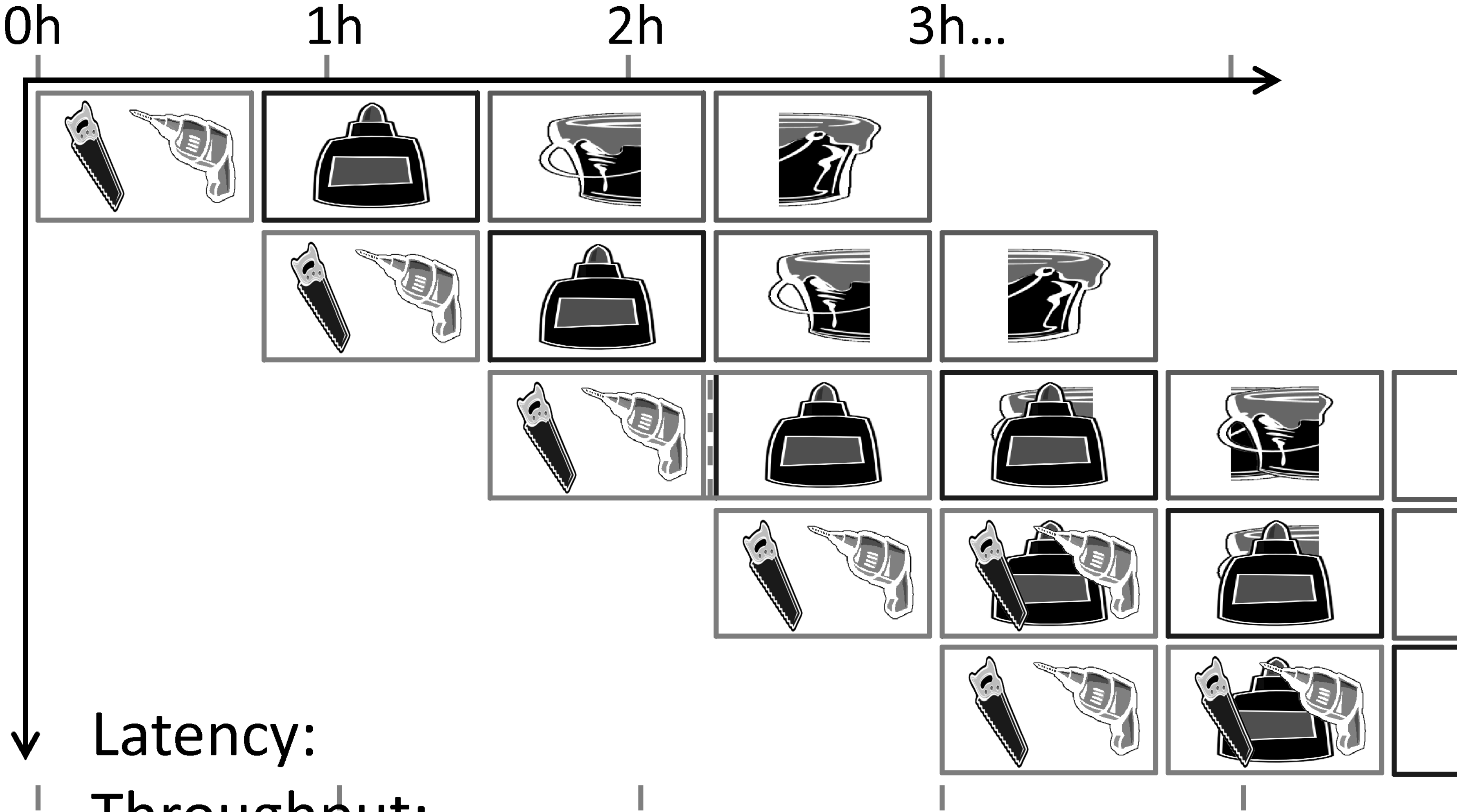


Latency:

Throughput:

Concurrency:

## Q: What if glue step of task 3 depends on output of task 1?



Latency:

Throughput:

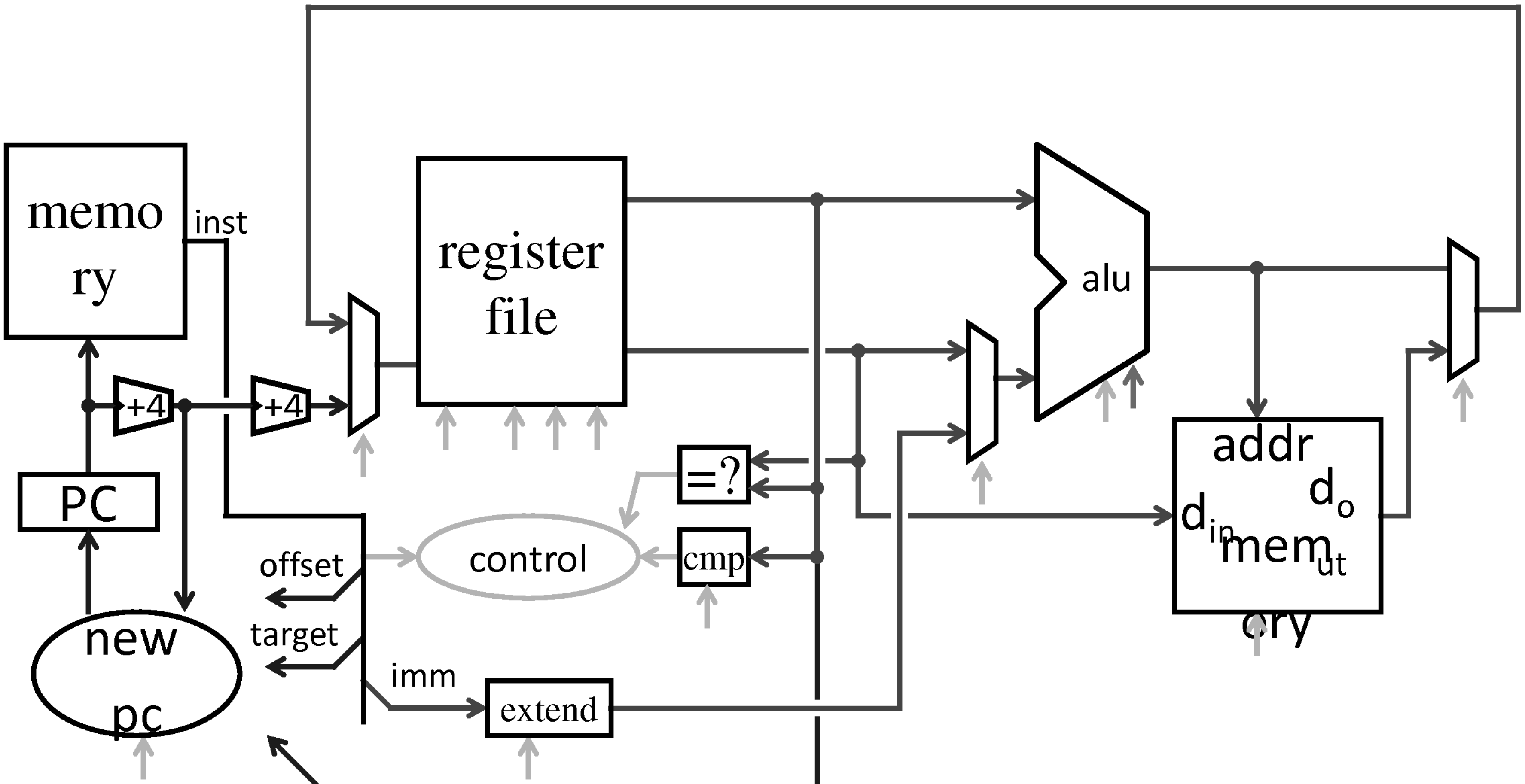
Concurrency:

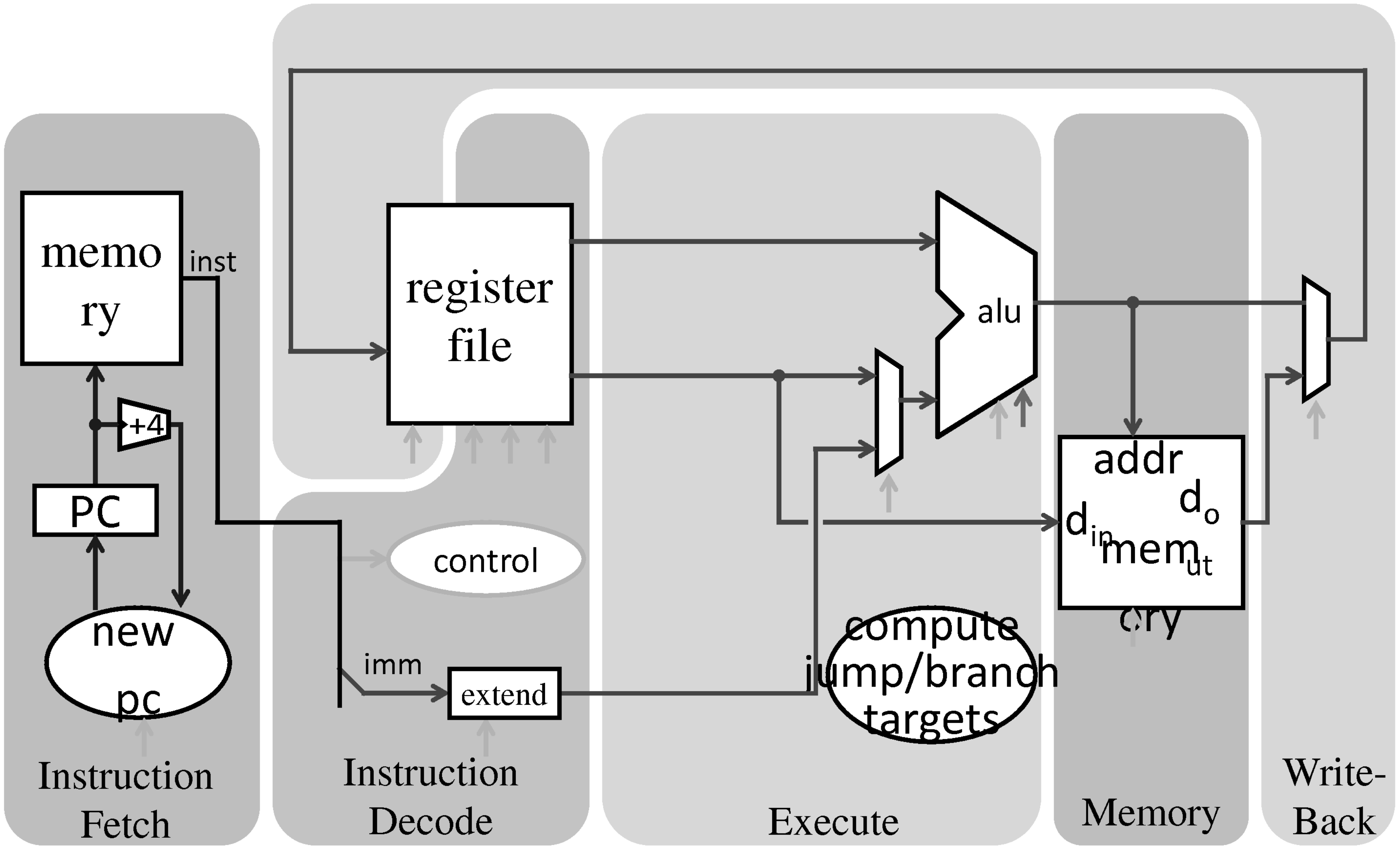
## Principle:

Latencies can be masked by parallel execution

## Pipelining:

- Identify *pipeline stages*
- Isolate stages from each other
- Resolve pipeline *hazards*


memo
ry
inst
register
file
alu
PC
+4
+4
=?
cmp
control
offset
target
imm
extend
new
pc
addr
$d_o$
$d_{in}$
mem
ut
ory


memory
inst
register file
alu
+4
PC
new pc
control
imm
extend
addr
$d_{in}$
$d_{out}$
memory
compute jump/branch targets
Instruction Fetch
Instruction Decode
Execute
Memory
Write-Back

## Five stage "RISC" load-store architecture

1. Instruction fetch (IF)
   – get instruction from memory, increment PC
2. Instruction Decode (ID)
   – translate opcode into control signals and read registers
3. Execute (EX)
   – perform ALU operation,  compute jump/branch targets
4. Memory (MEM)
   – access memory if needed
5. Writeback (WB)
   – update register file

Slides thanks to Sally McKee & Kavita Bala

Break instructions across multiple clock cycles (five, in this case)

Design a separate stage for the execution performed during each clock cycle

Add pipeline registers to isolate signals between different stages