# Simple Scalar Processor

Ainsley Rutterford

ar16478@bristol.ac.uk

# Instruction Set

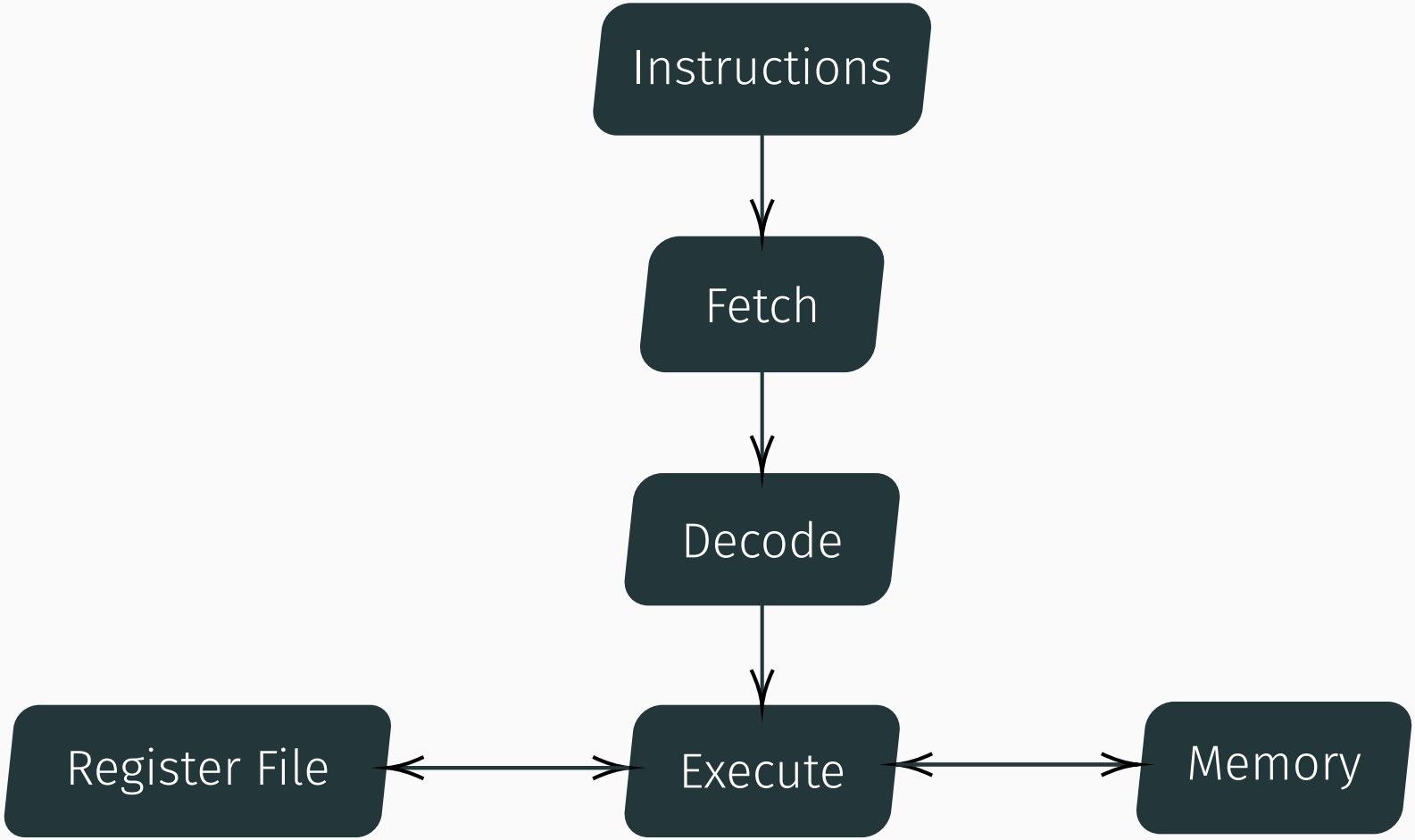
```
nop
add  dst src1 src2
addi dst src1 const
sub  dst src1 src2
subi dst src1 const
mul  dst src1 src2
div  dst src1 src2
lw   dst base(id)
li   dst const
sw   base(id) src
```

```
beq  r1 r2 dst
bne  r1 r2 dst
bgt  r1 r2 dst
bge  r1 r2 dst
blt  r1 r2 dst
ble  r1 r2 dst
j    dst
jal  dst
jr   r1
```

# Benchmarks

| | |
|---|---|
| Bubblesort | Sort an array of 20 integers. |
| Iterative GCD | Calculate GCD of two integers iteratively. |
| Recursive GCD | Calculate GCD of two integers recursively. |
| Vector addition | Add two vectors of length 20. |

# Architecture Overview



# Notes and Questions

Written in Python and works with `python3` available on the lab machines.

Right now it is non-pipelined, scalar, and executes instructions in-order.

Currently, the fetch, decode, and execute stages take 1 cycle each. So it takes 3 cycles for an instruction to be executed.

Later on, would it be sensible to arbitrarily say that loads and stores take 3 cycles while arithmetic takes 1 cycle for example?

Would it be sensible to add a write back stage after execute?