Digital Electronics and Microprocessor Systems (ELEC211)

Dave McIntosh and **Valerio Selis**

dmc@liverpool.ac.uk
V.Selis@liverpool.ac.uk

UNIVERSITY OF LIVERPOOL

Outline

Instruction pipelines

- Revision
- Branch instruction
- Load and store instructions

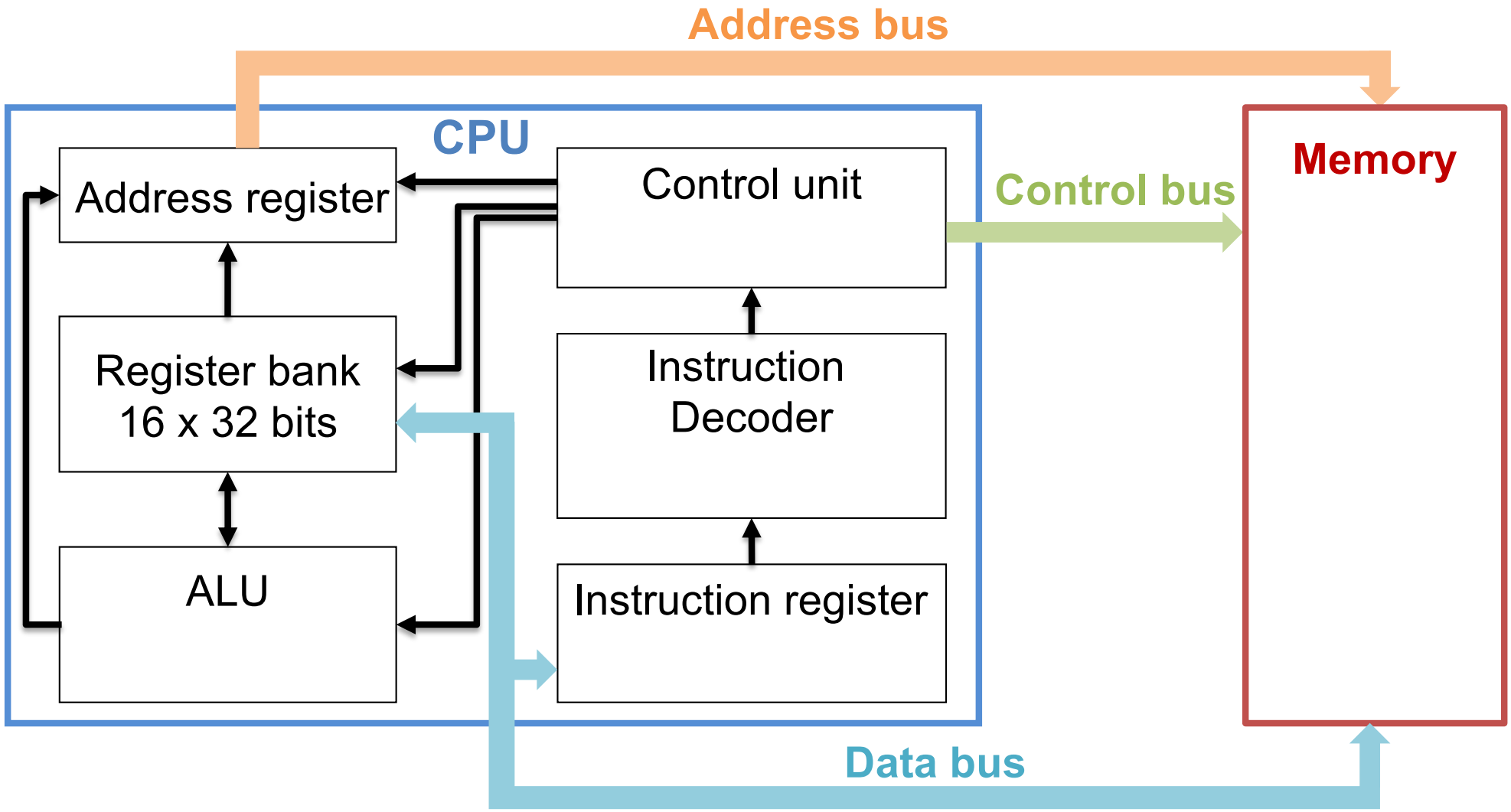
Von Neumann VS Harvard Architectures

- ARM7 VS ARM9

Interrupts

- Interrupt handling
- Interrupt request (IRQ) mode

CPU and memory

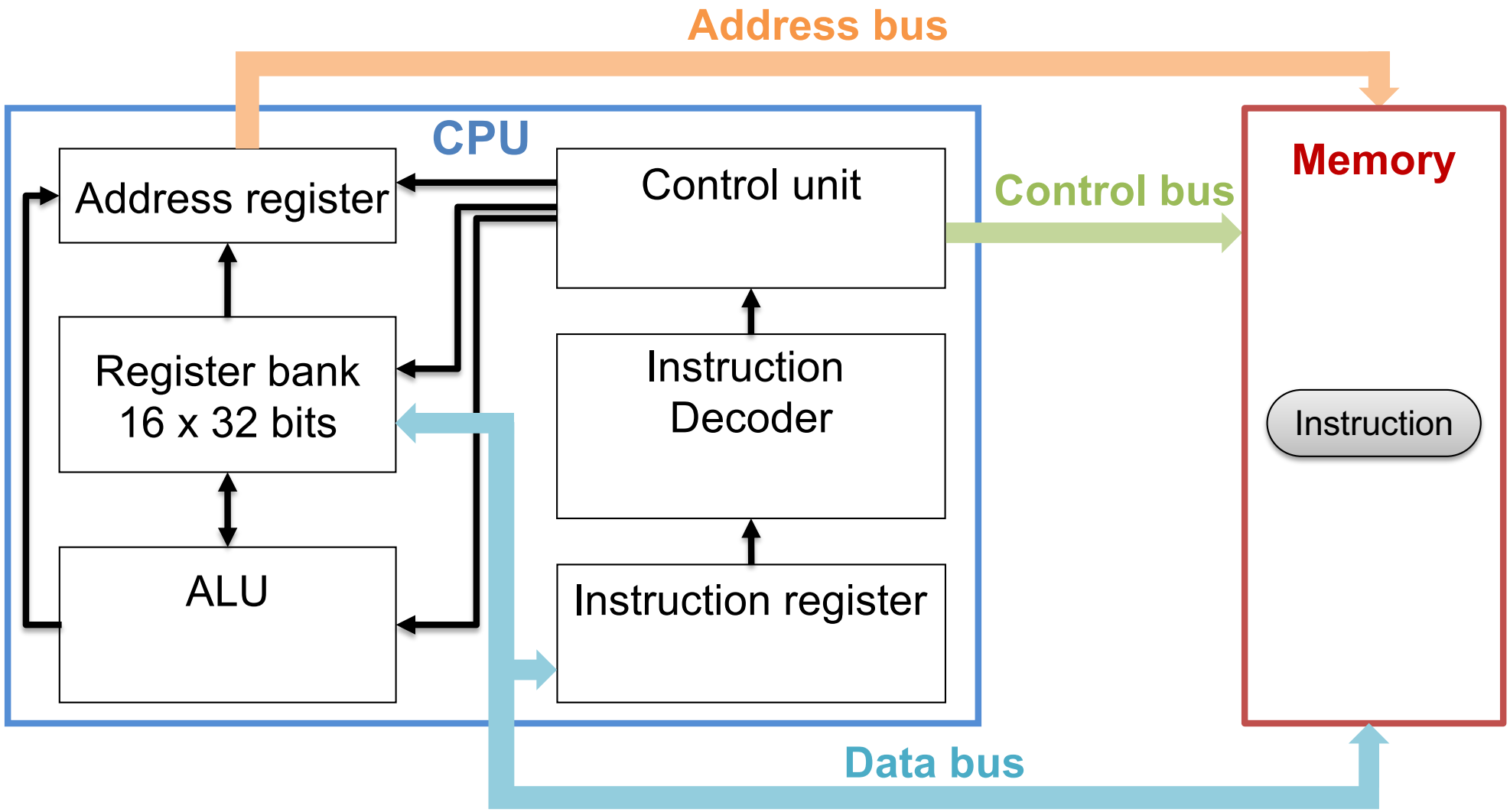


Fetch, decode, execute

The CPU performs three cycles sequentially:

- During the **FETCH** cycle an instruction in memory is loaded into the instruction register.

Fetch cycle

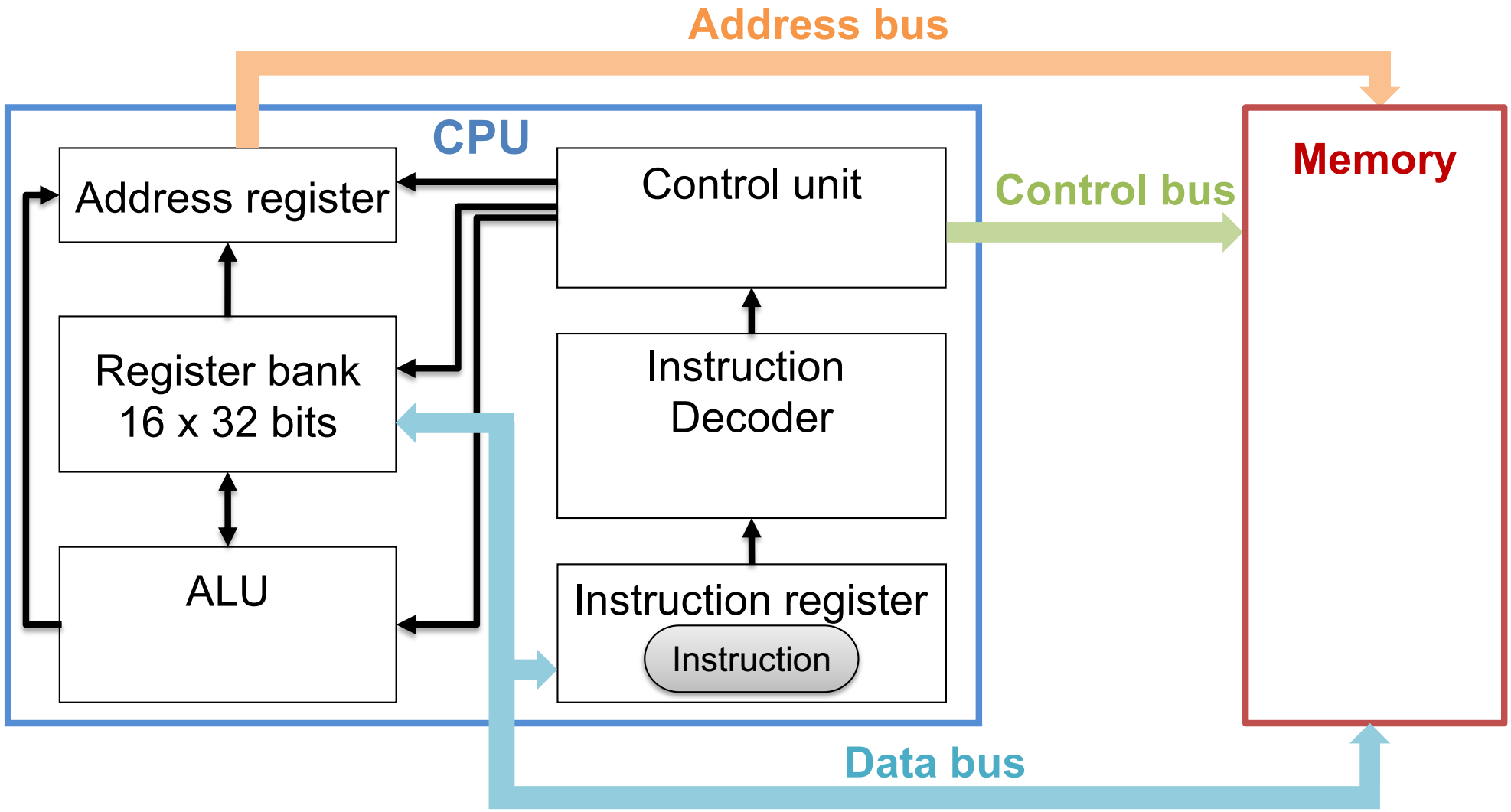

Address bus
CPU
Address register
Control unit
Control bus
Memory
Register bank
16 x 32 bits
Instruction
Decoder
Instruction
ALU
Instruction register
Data bus

Fetch, decode, execute

The CPU performs three cycles sequentially:

- During the **FETCH** cycle an instruction in memory is loaded into the instruction register.
- During the **DECODE** cycle the instruction is interpreted by the instruction decoder.

Decode cycle

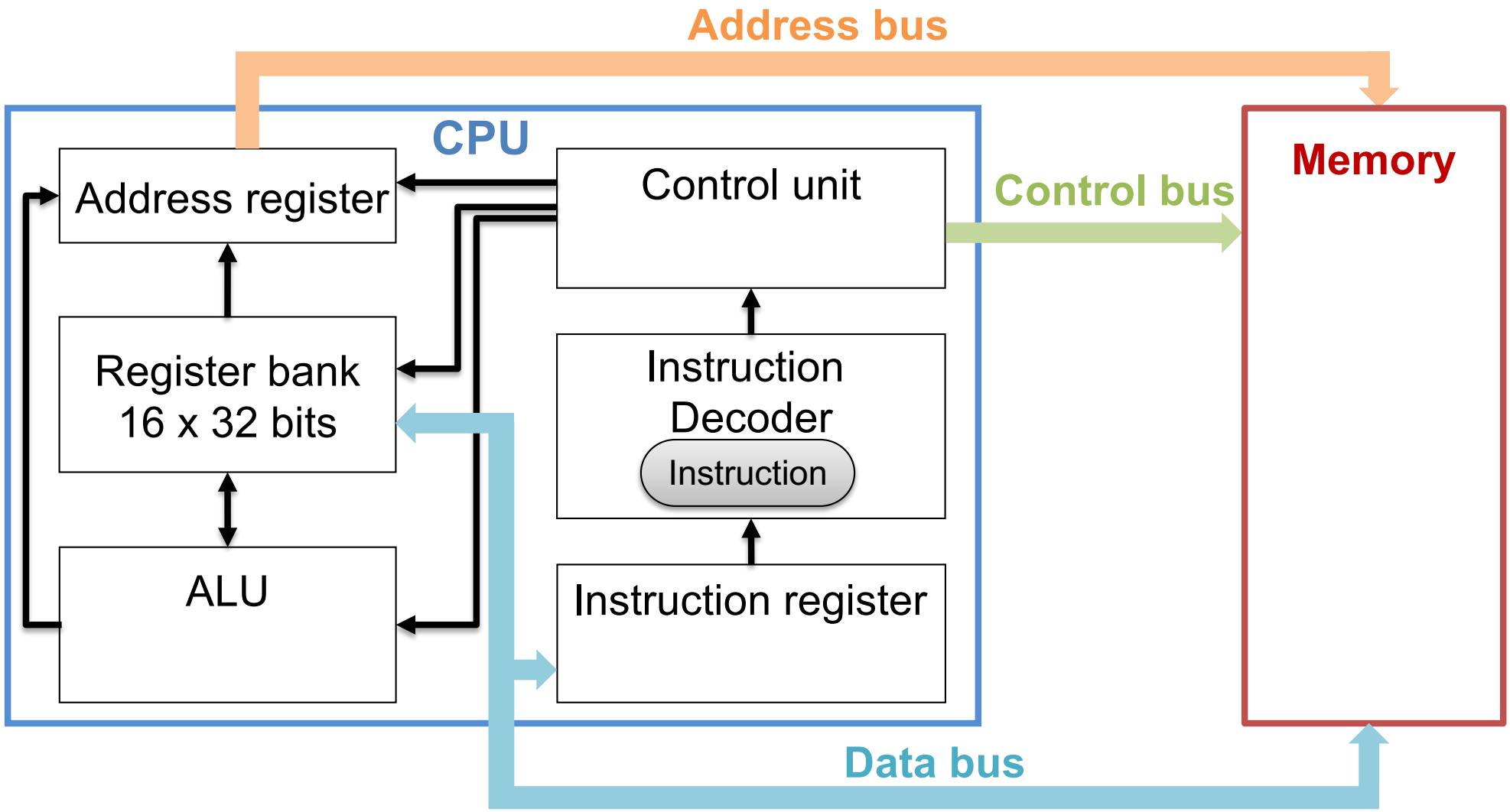


Fetch, decode, execute

The CPU performs three cycles sequentially:

- During the **FETCH** cycle an instruction in memory is loaded into the instruction register.
- During the **DECODE** cycle the instruction is interpreted by the instruction decoder.
- During the **EXECUTE** cycle either the ALU performs a calculation on values held in registers,
- or a value in a register is stored into memory,
- or a value in memory is loaded into a register.

Execute cycle



Instruction Pipelines

Instruction pipelines are an important feature of all modern microprocessors.

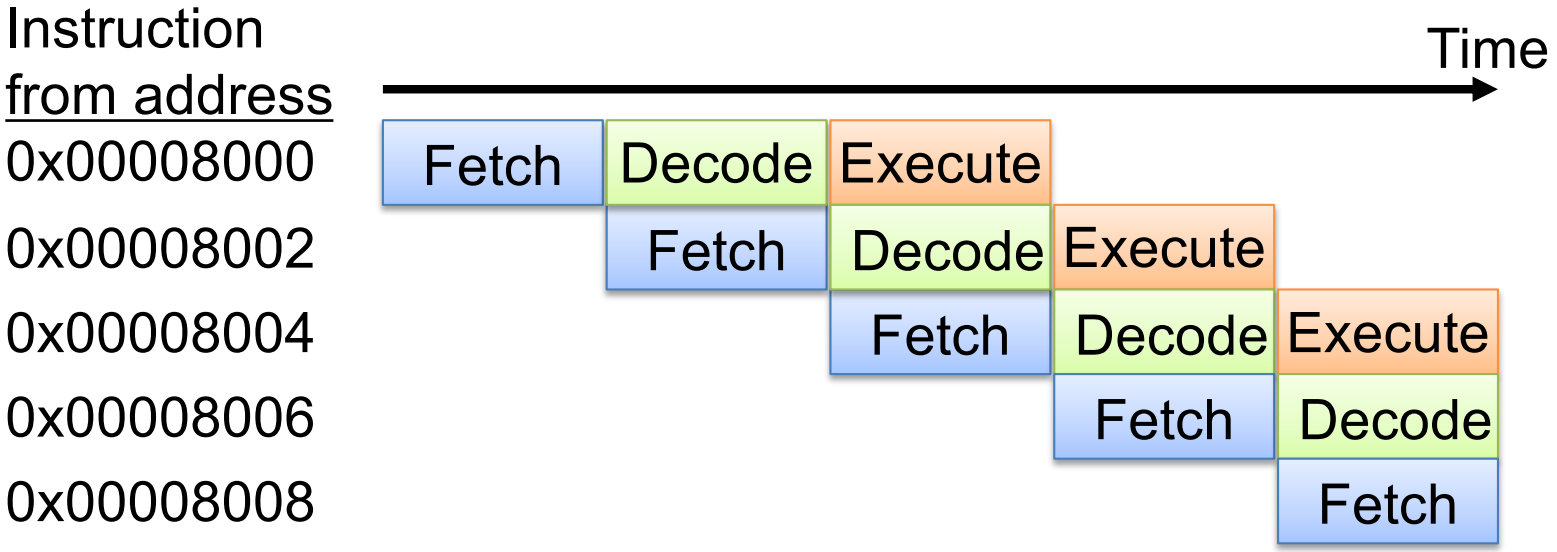
For the same basic speed of transistor operation, an *n* stage instruction pipeline allows the microprocessor to execute up to *n* times as many instructions in a given time.

The ARM Cortex M0 microprocessor has a **three stage pipeline**:

- Fetch, decode and execute.

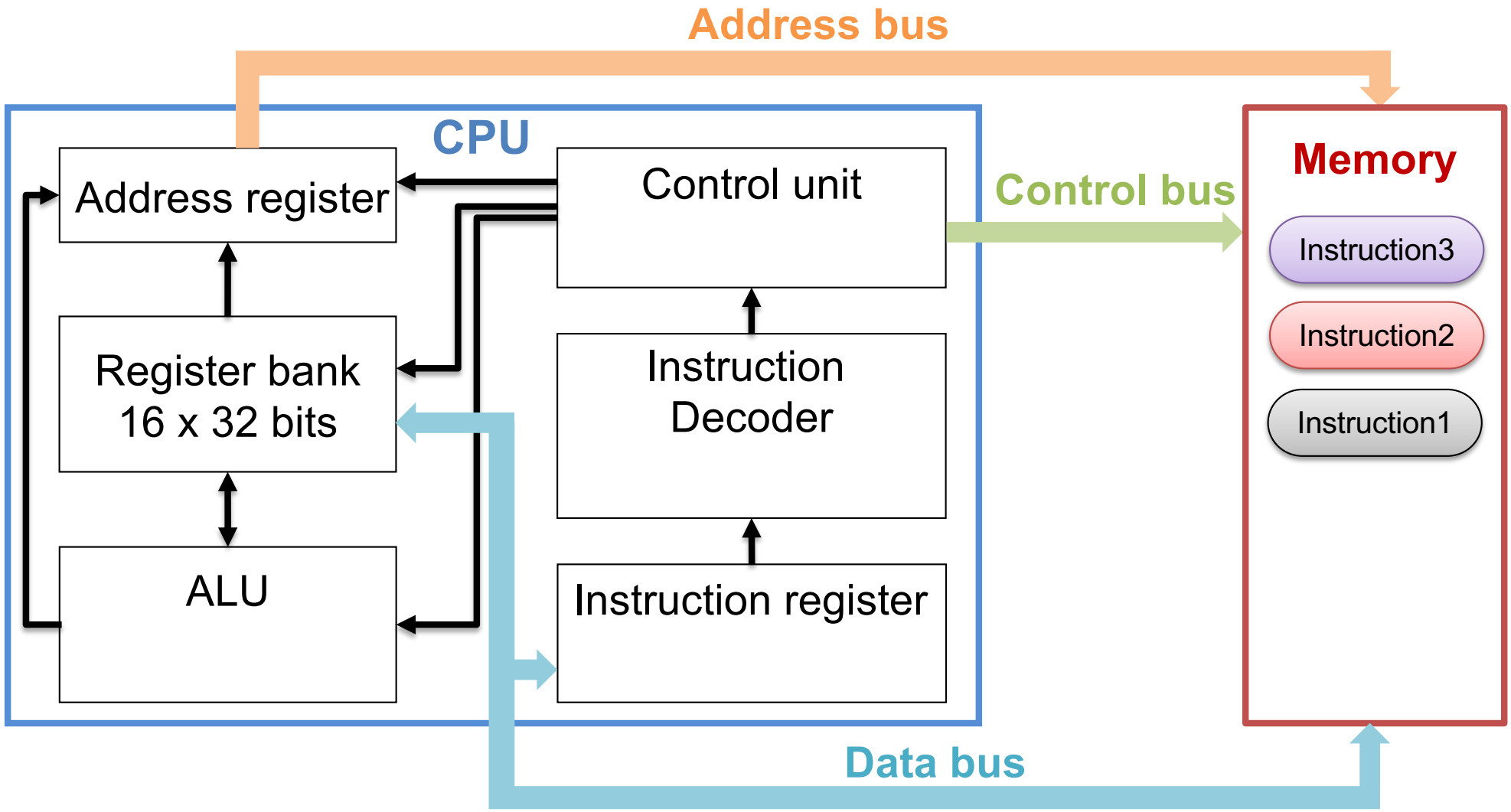
Instruction Pipelines

In a three stage pipeline, the CPU can simultaneously **execute** an instruction, **decode** the next instruction and **fetch** the next but one instruction.

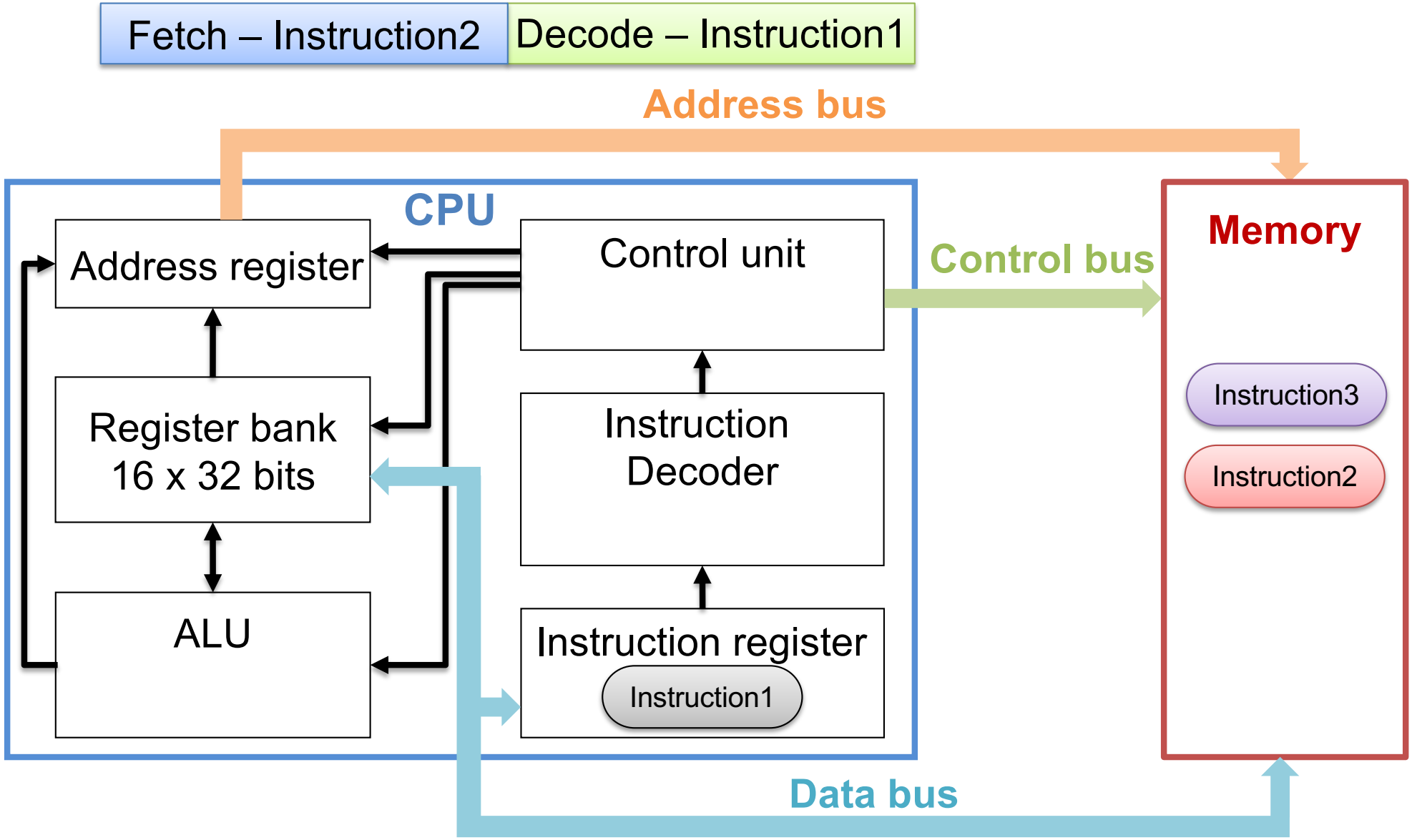


Instruction Pipelines

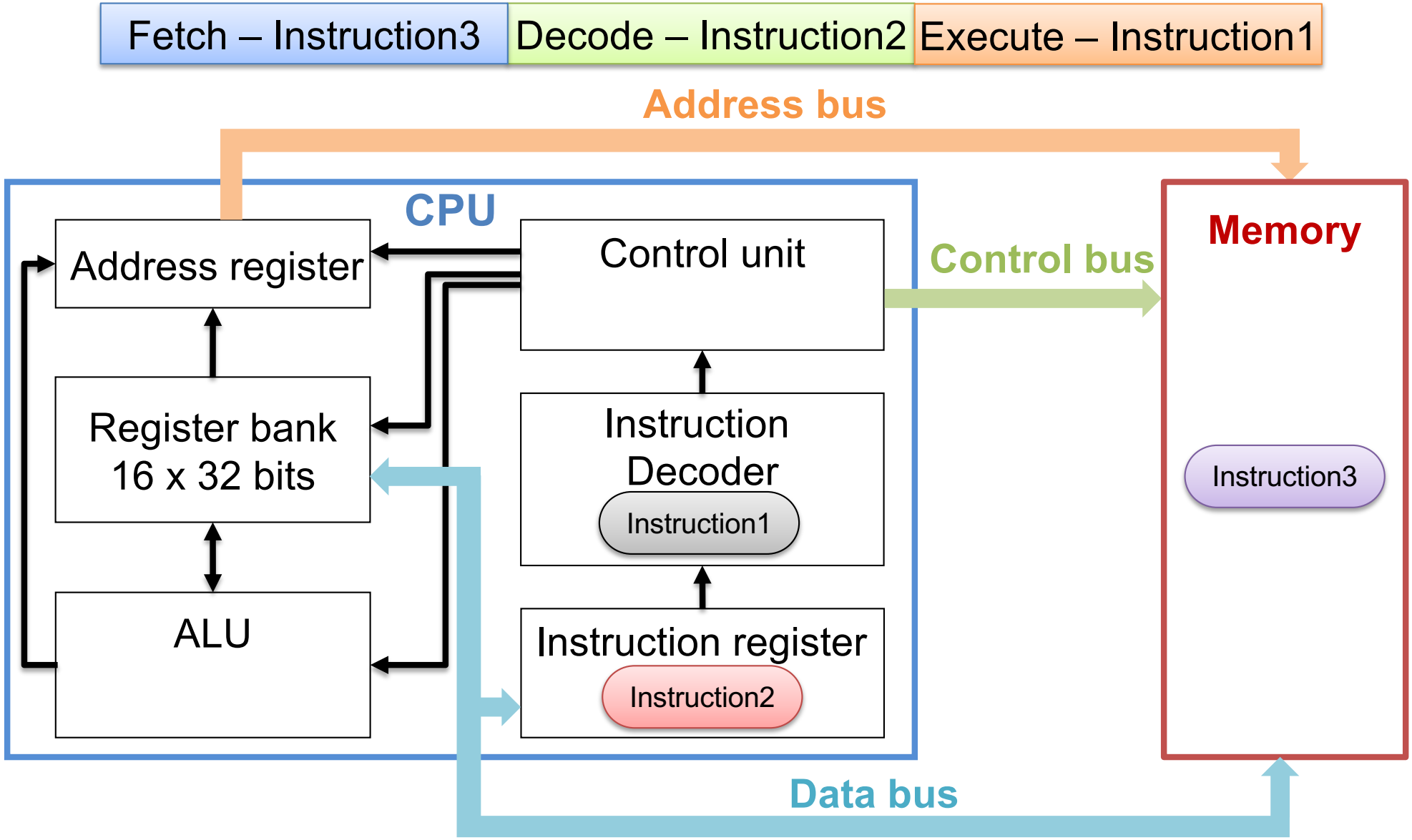
Fetch – Instruction1



Instruction Pipelines



Instruction Pipelines



Pipelines - optimum operation

A pipeline operates optimally if:

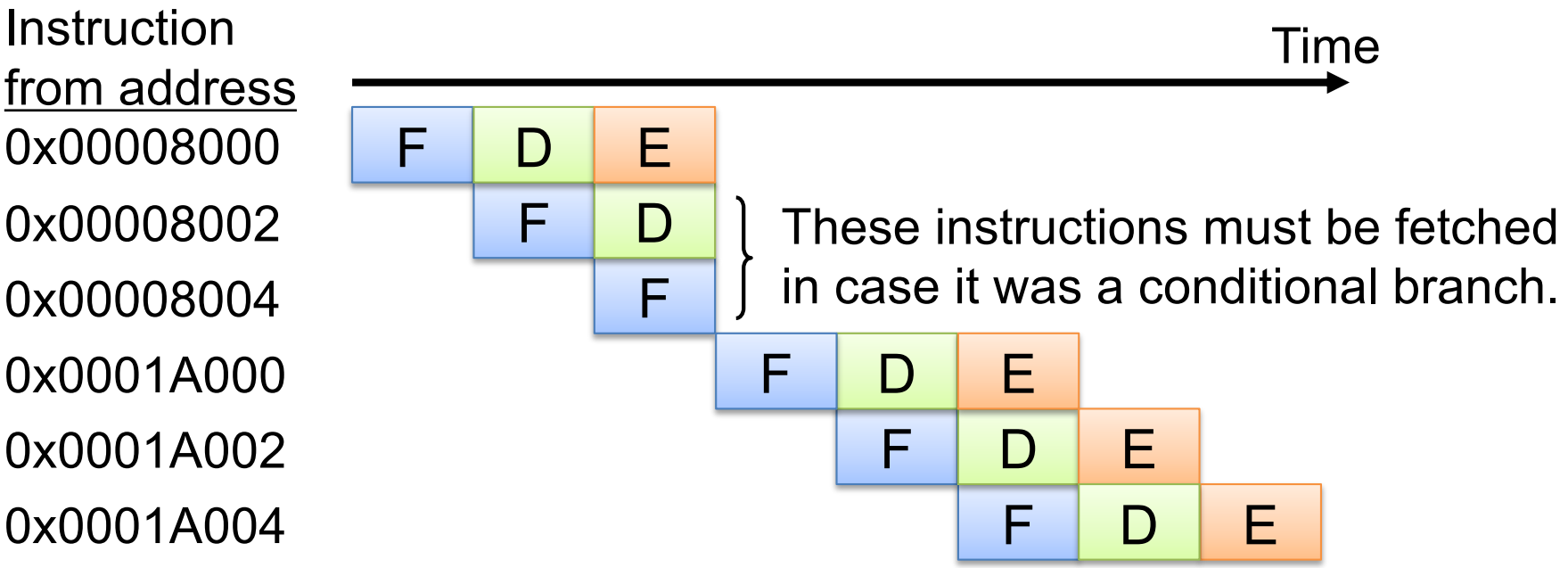
- The instructions to be executed are in consecutive memory locations,
- No conflict occurs on the data bus

The microprocessor will operate at one clock cycle per instruction (1 CPI).

The best performance cannot be achieved if a branch, load or store instruction is executed or if an interrupt is serviced.

Branch Instruction

A branch instruction will reload the program counter so that two clock cycles are lost e.g. assume that the instruction at address 0x00008000 is Branch to 0x0001A000



Pipelines - optimum operation

So in **six** clock cycles only **four** instructions are **executed**:

- 1.5 clock cycles per instruction (CPI = **6**/**4** = 1.5).

For a branch and link instruction, the link register is updated during the two clock cycles when no instruction is being executed.

If a conditional branch is not executed then no clock cycles are lost.

Question

```
        MOVS r0, r0
        BNE cont                ; branch if not zero
        ADDS r3, r4, r5
        EORS r2, r1, r4
cont    SUBS r6, r7, r2
```

When poll is active, respond at **PollEv.com/elec211**

Text **ELEC211** to **22333** once to join

How many clock cycles do the instructions take to execute if the value in r0 is (a) not equal to 0 and (b) equal to 0? What is the CPI?

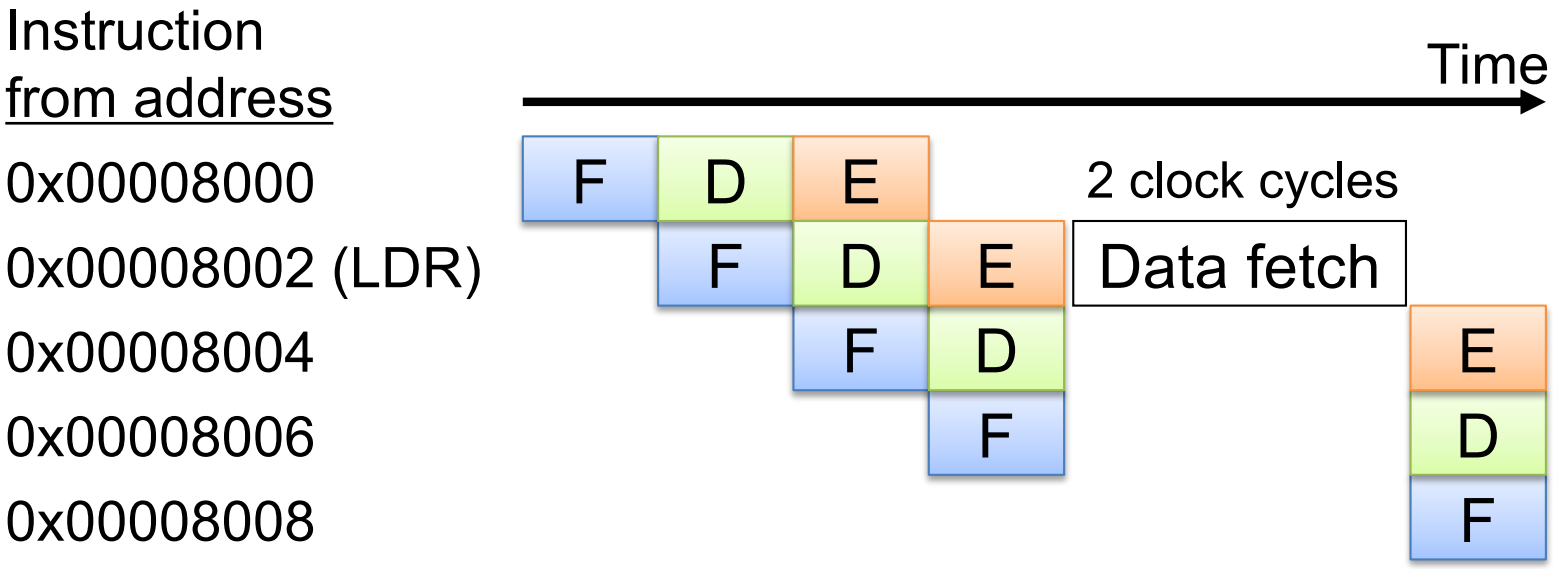
- (a) 3 and 1 CPI; (b) 5 and 1 CPI
- (a) 5 and 1.66 CPI; (b) 5 and 1 CPI
- (a) 5 and 1 CPI; (b) 3 and 1 CPI
- (a) 5 and 1 CPI; (b) 5 and 1.66 CPI

Total Results: 0

Start the presentation to see live content. Still no live content? Install the app or get help at **PollEv.com/app**

Load and store instructions

In the ARM Cortex M0 CPU **load and store** instructions use the same data bus to pass data to or from memory so that the data bus cannot be used to fetch an instruction at the same time.



Eliminating bus conflicts

The data fetch uses two clock cycles so that in the previous example only three instructions are executed in five clock cycles - 1.66 CPI.

Bus conflicts could be eliminated by using **two separate data buses**:

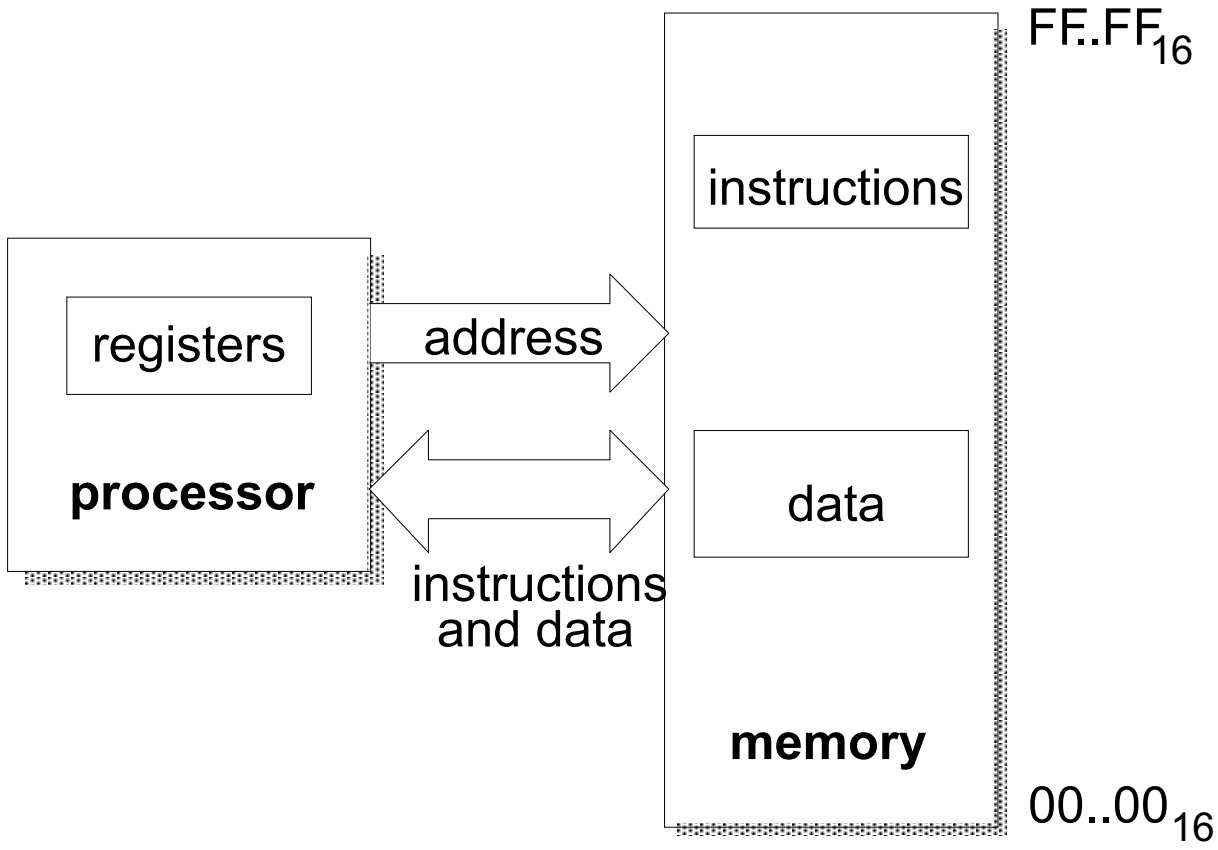
- One for fetching instructions
- One for loading and storing data

This is called a **Harvard architecture**.

A microprocessor, e.g. the ARM Cortex M0, that uses **one data bus** for both instructions and data is said to have the **von Neumann architecture**.

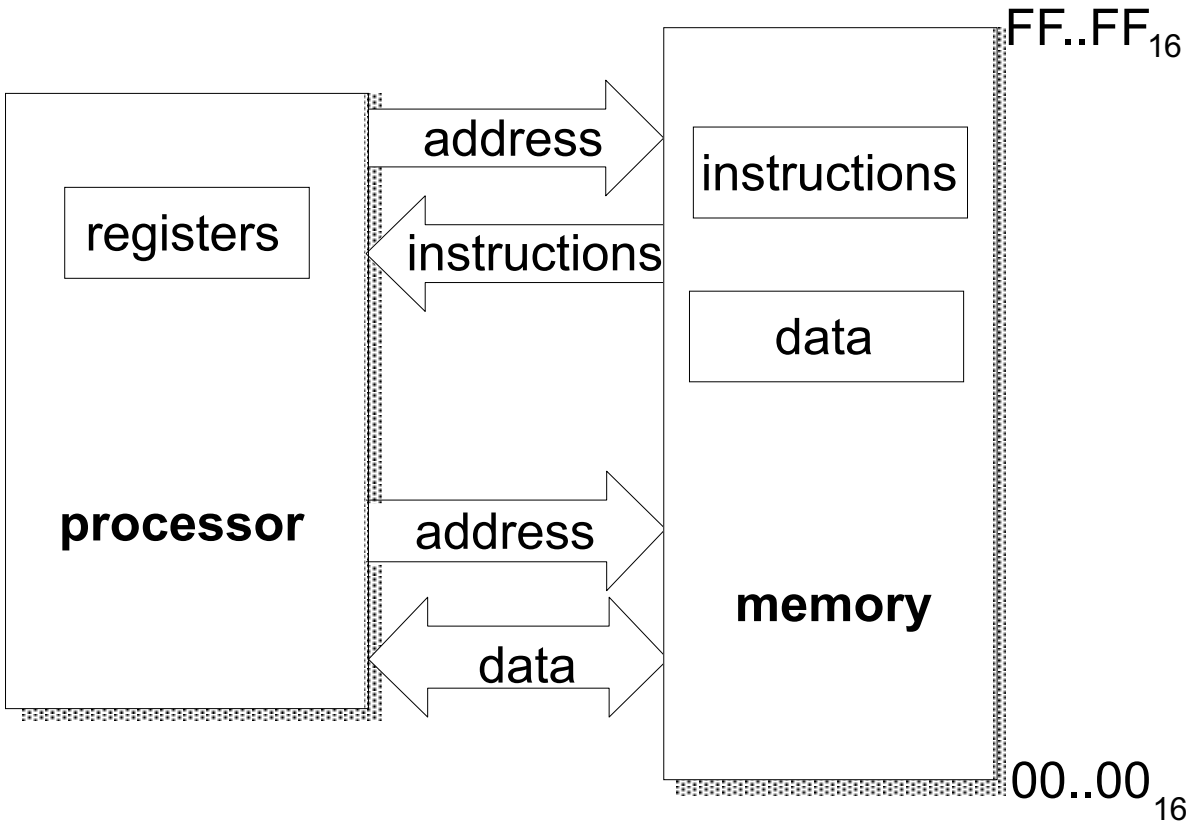
Von Neumann architecture

One data bus and one address bus used for both instructions and data.



Harvard architecture

Separate data and addresses busses – one pair only for data and the other pair only for instructions.



Von Neumann or Harvard?

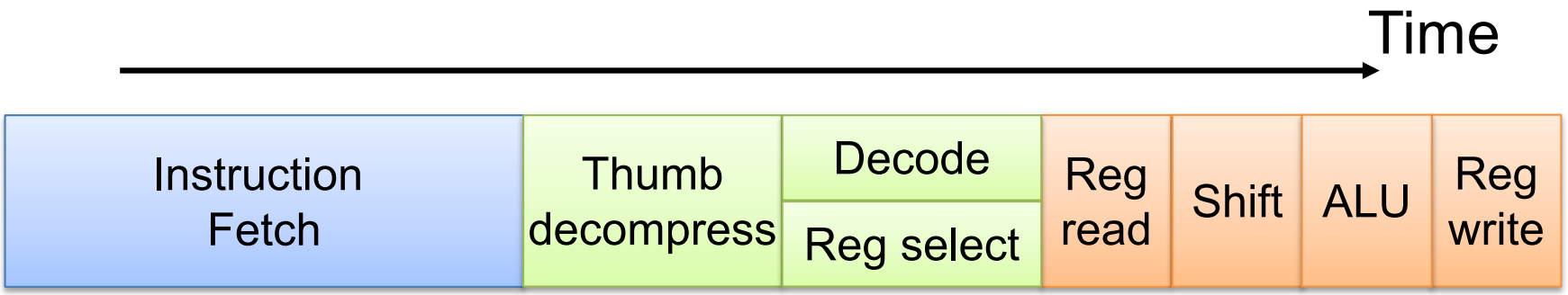
The advantage of Harvard over von Neumann is that there are fewer lost clock cycles due to bus conflicts. The drawback is greater complexity.

The ARM9 (next generation after ARM7) has a Harvard architecture & a five stage pipeline.

The ARM7 has an average CPI of 1.9 whereas the ARM9 has an average CPI of 1.5. Note that this depends upon the program being executed but typical data processing programs make a lot of use of load and store.

ARM7 3 stage pipeline: detail

In each stage of the ARM7 pipeline several things happen; normally consecutively:



FETCH

Instruction fetched from memory.

DECODE

Thumb to ARM decompression.
Decode.
Register selection.

EXECUTE

Register read.
Barrel shift.
ALU operation.
Register write.

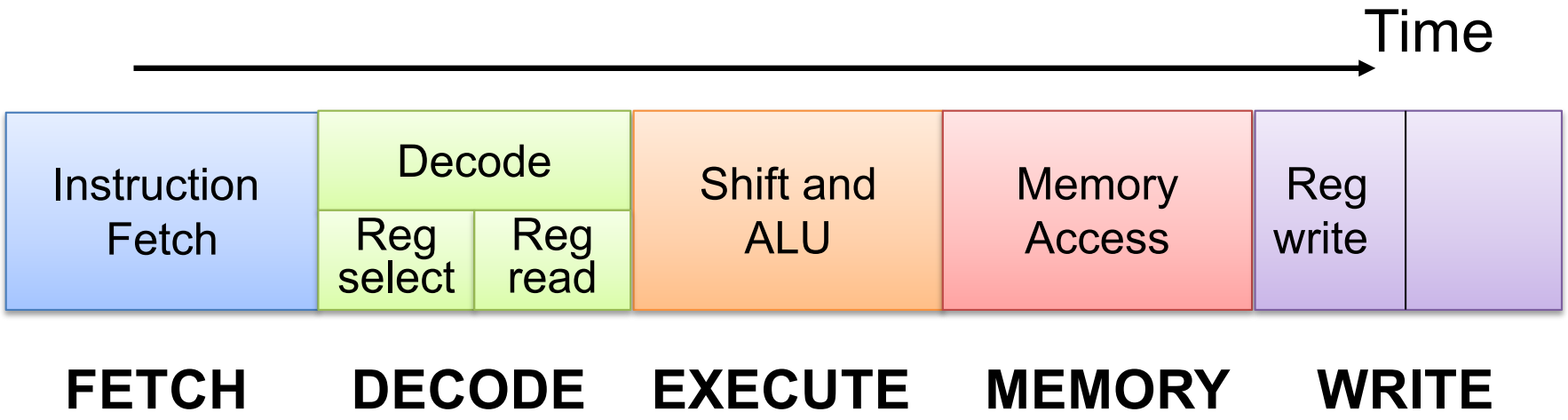
3 stage or 5 stage pipeline?

The maximum clock frequency of the ARM7 is limited by a bottleneck during the execute stage of the 3 stage pipeline.

The ARM9 has a 5 stage pipeline so that there is less work in each stage of the pipeline and therefore a higher maximum clock frequency can be achieved for the same technology.

However more stages require more power because more circuits are functioning simultaneously.

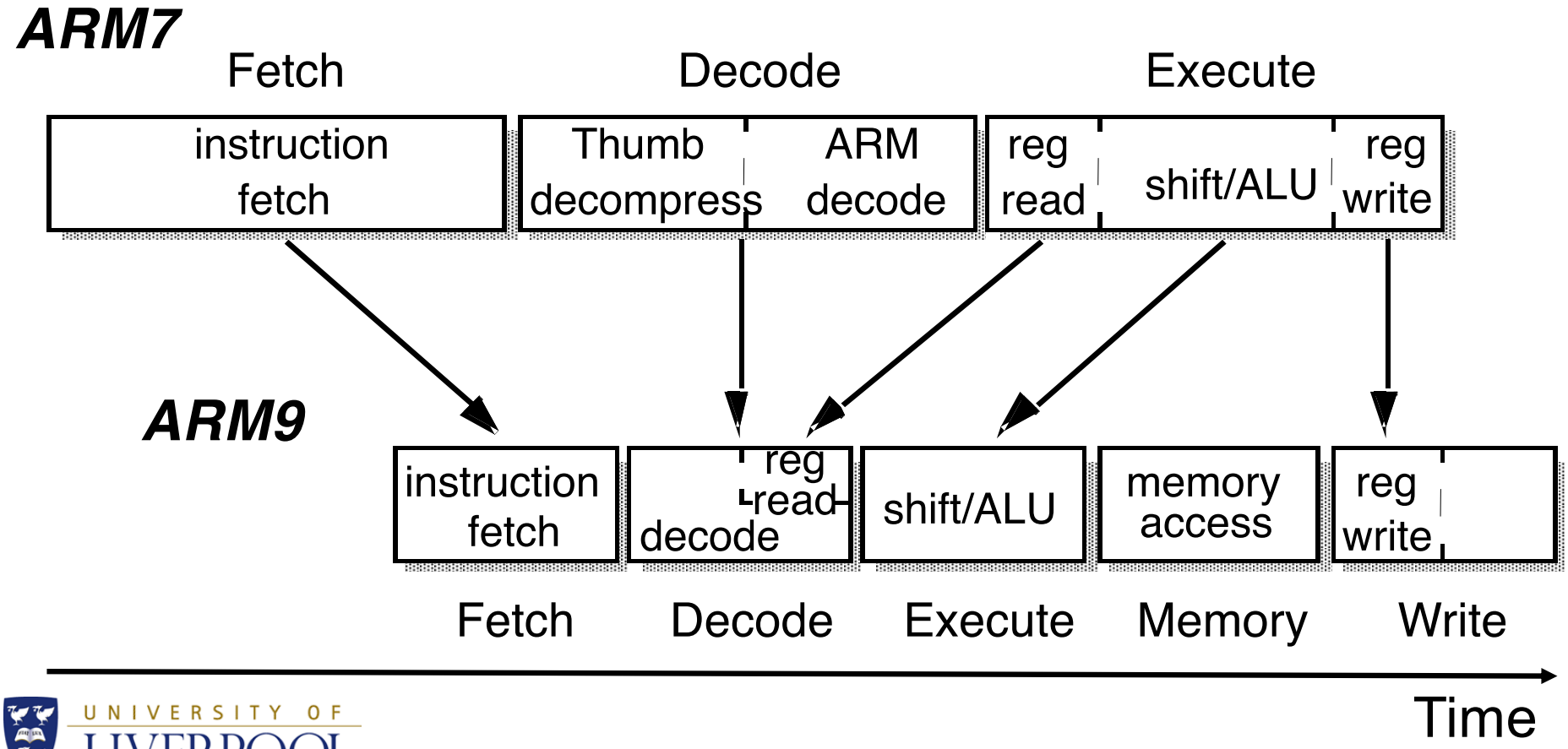
ARM9 5 stage pipeline: detail



Register read is moved into the decode stage and register write is delayed until after a separate stage for any memory data access (load/store).

ARM7 & ARM9 pipeline comparisons

More stages = less activity per stage
= higher clock frequency
= higher performance



ARM7:ARM9 comparison

Processor:	ARM720T	ARM922T
Architecture:	Von Neumann	Harvard
Pipeline:	3 stage	5 stage
Transistors in core:	74,209	111,000
Area including cache:	2.4 mm^2	3.2 mm^2
Clock frequency:	0 to 100 MHz	0 to 250 MHz
Power:	upto 20 mW	upto 62.5 mW
mW/MHz:	0.2	0.25

The ARM720T has a 8 kB unified cache memory (one cache for data and instructions) whereas the ARM922T has a 8 kB instruction cache and a 8 kB data cache.

Interrupts

Many applications for microprocessor systems require things to happen when **an event occurs unexpectedly**.

These unexpected events can interrupt the normal operation of the microprocessor.

During an ‘interrupt’, the execution of the main program is halted and a special ‘interrupt’ program is executed instead. When the interrupt program has finished, the microprocessor returns to the main program.

Interrupts

Interrupts provide a very convenient method for dealing with events and this method is often used for events that are not unexpected e.g. when a mobile phone receives an incoming call.

An interrupt is triggered when the microprocessor receives a (voltage) signal on a special connection within the control bus.

The ARM Cortex M0 has two types interrupts:

- Standard interrupts (IRQ) and one
- Non-maskable interrupt (NMI) – highest priority.

Interrupt Handling

What happens when the microprocessor receives an interrupt signal?

The microprocessor switches ‘mode’. The ARM normally operates in ‘**user mode**’ - when a normal interrupt is received it switches to ‘**IRQ mode**’.

Each mode has its own link register and stack pointer.

Interrupt Handling

When an interrupt occurs the following happens:

1. Some of the registers in the register bank (r0 to r3, r12, r14), the return address (pc), and the Current Program Status Register (CPSR) are **pushed** to the current active stack memory.
2. The return address of the next instruction to be executed in the main program is stored in the link register for the appropriate mode.
3. The program counter is set to the correct **exception handler address** for that IRQ starting from 0x00000040 to 0x00000040+4*n* (IRQ0 to IRQ*n*). These memory addresses are known as '**vectors**'.

Returning from Interrupts

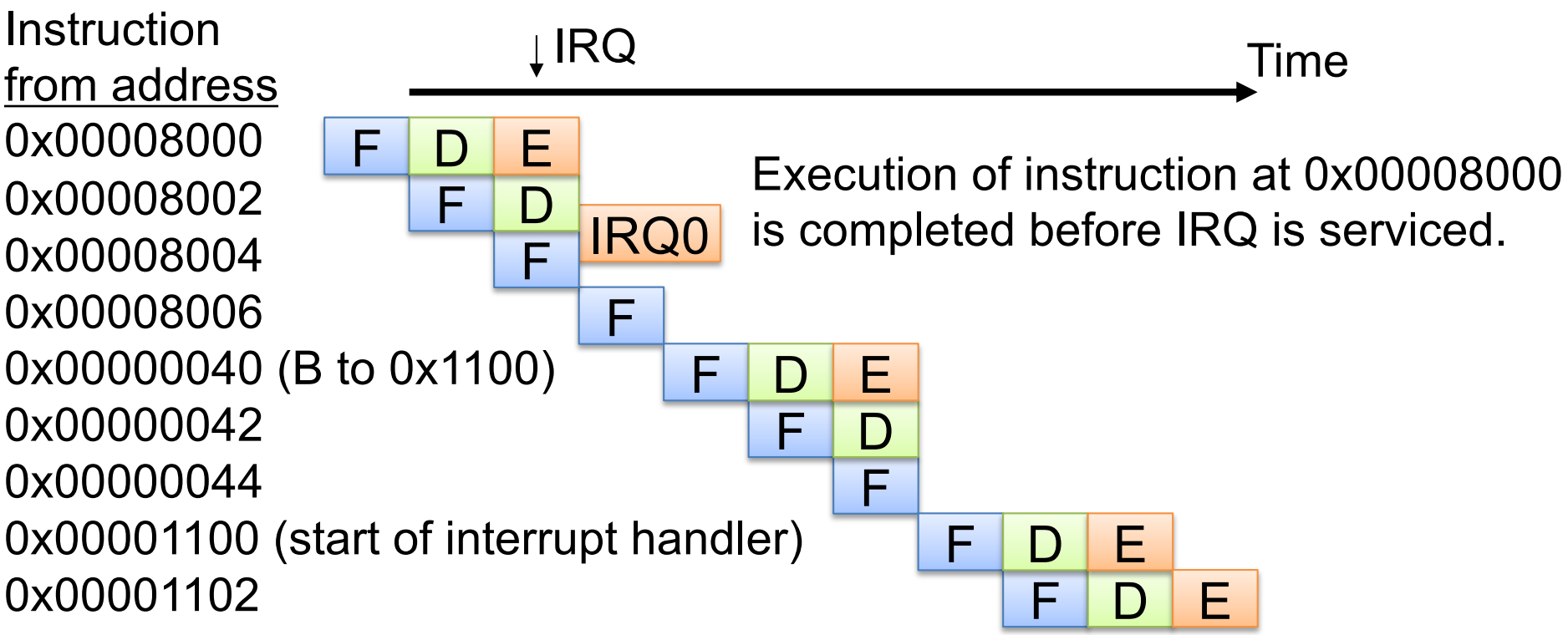
When the program called by the interrupt has finished the microprocessor returns to the main program.

This is achieved by doing the following:

1. The saved program status register is copied back into the current program status register (CPSR).
2. The link register in the IRQ mode is copied into program counter (r15) and the other registers are **popped** back into the register bank.
3. The microprocessor returns to the mode it was in before the interrupt occurred - normally user mode.

Interrupts

An IRQ0 reloads the PC with 0x00000040 and then branches so that the pipeline is broken twice.



Recommended reading

J. R. Gibson, ARM Assembly Language – an Introduction (2nd edition) ISBN 978-1-4477-1715-7

- Chapter 13 ARM hardware – for interrupts
- Section 3.3 Instruction execution – for pipelining

Summary

- Instruction pipelines
- Von Neumann architecture
- Harvard architecture
- Interrupts

Next class?

Wednesday at 12 noon in the
Chadwick building,
Barkla Lecture Theatre
(CHAD-BARKLA)