Digital Electronics and Microprocessor Systems (ELEC211)

Dave McIntosh and **Valerio Selis**

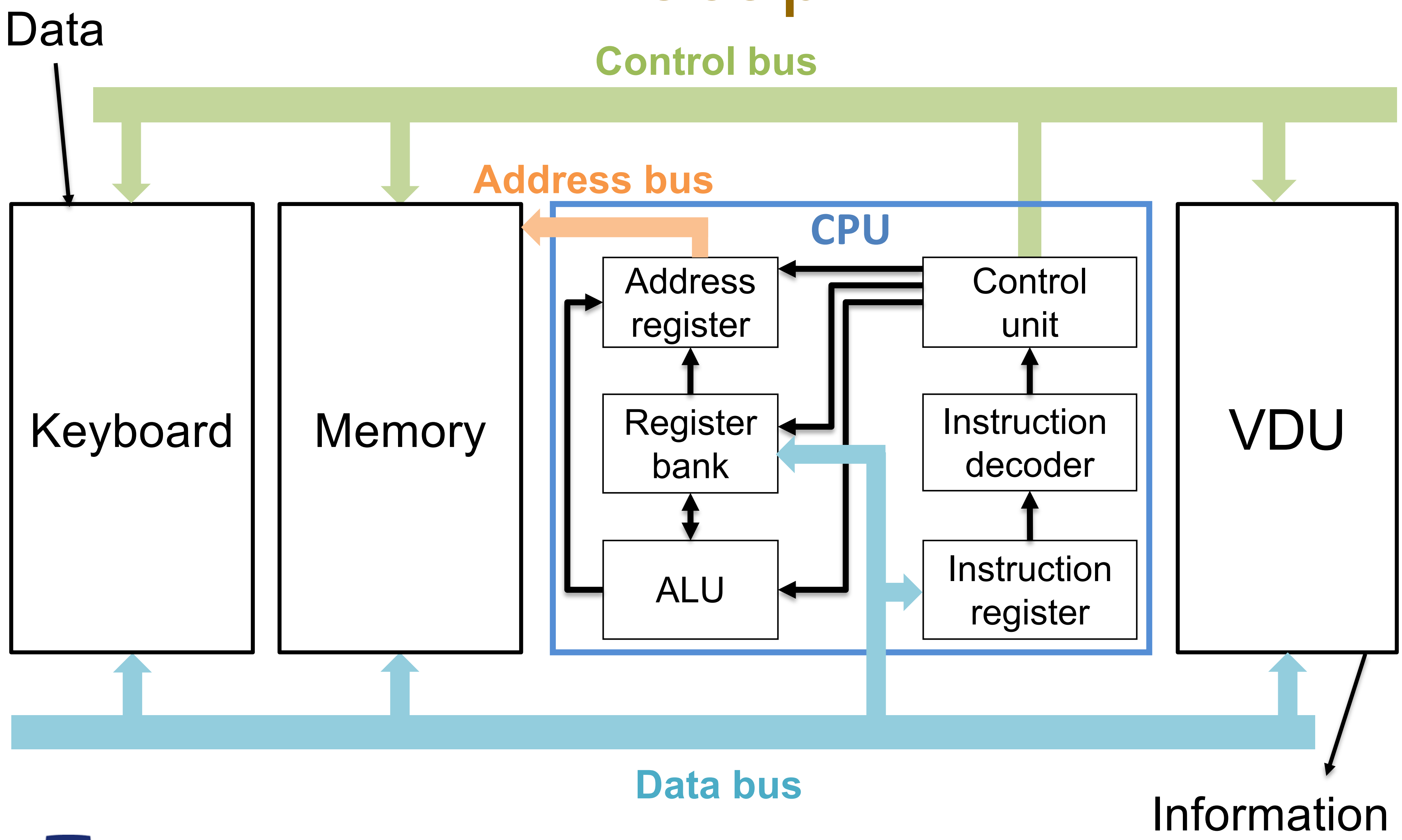
dmc@liverpool.ac.uk
V.Selis@liverpool.ac.uk

UNIVERSITY OF LIVERPOOL

Outline

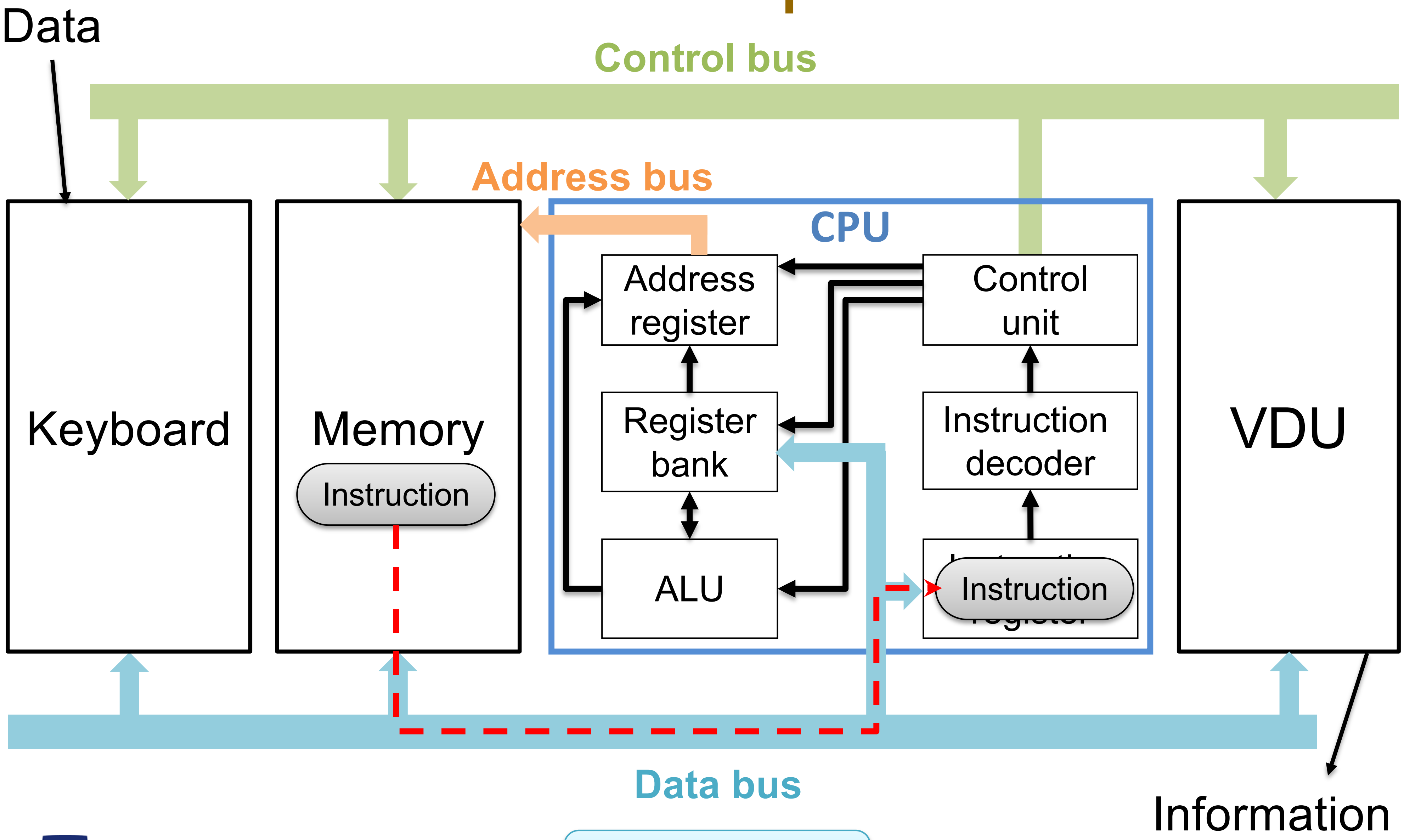
- Recap
- Simple instructions
- Mnemonics
- Assembly Language
    - Assembler
    - Compiler
- Types of instructions
    - Arithmetic Instructions

Recap



Data
Control bus
Address bus
CPU
Keyboard
Memory
Address register
Control unit
Register bank
Instruction decoder
ALU
Instruction register
VDU
Data bus
Information

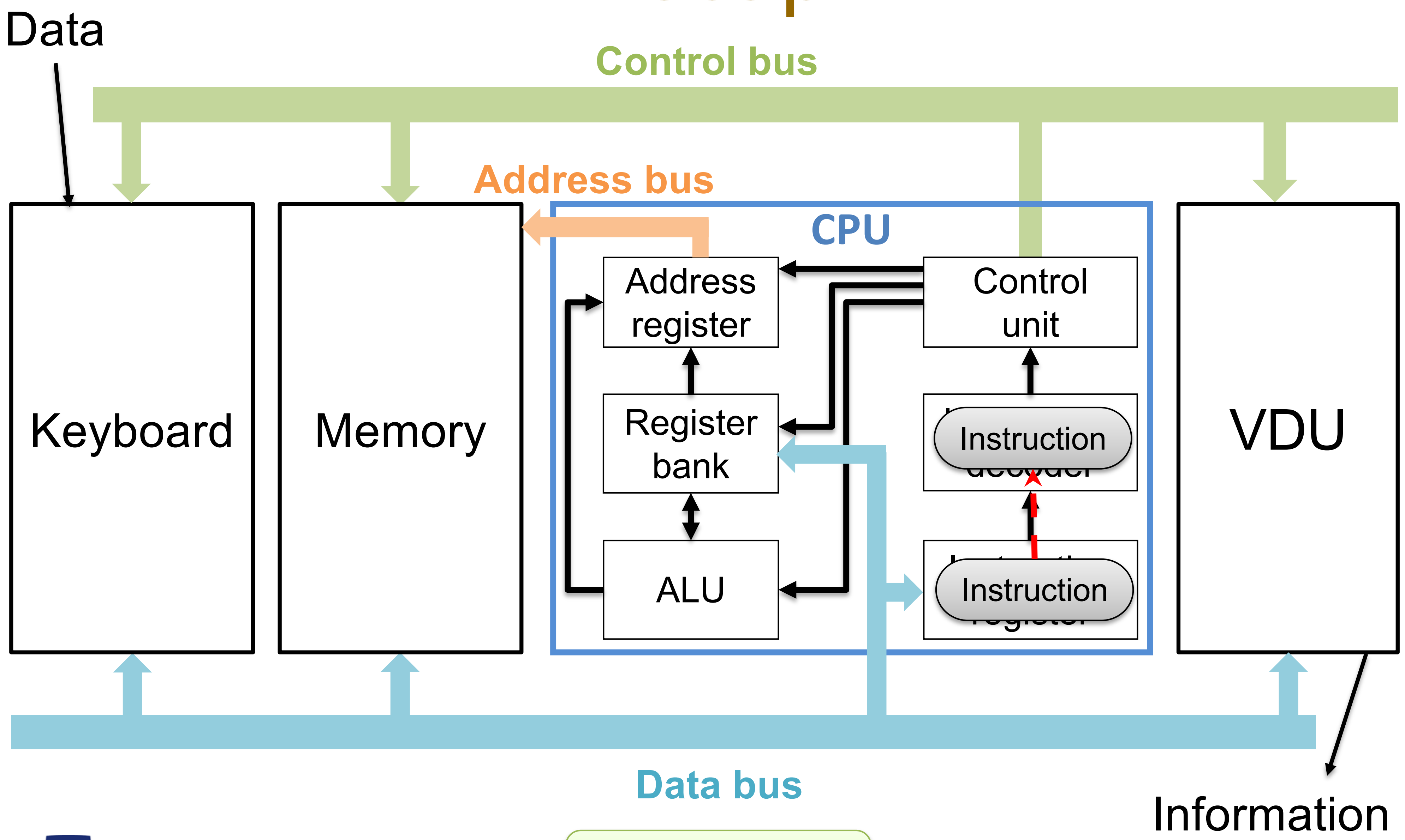
Recap



Fetch

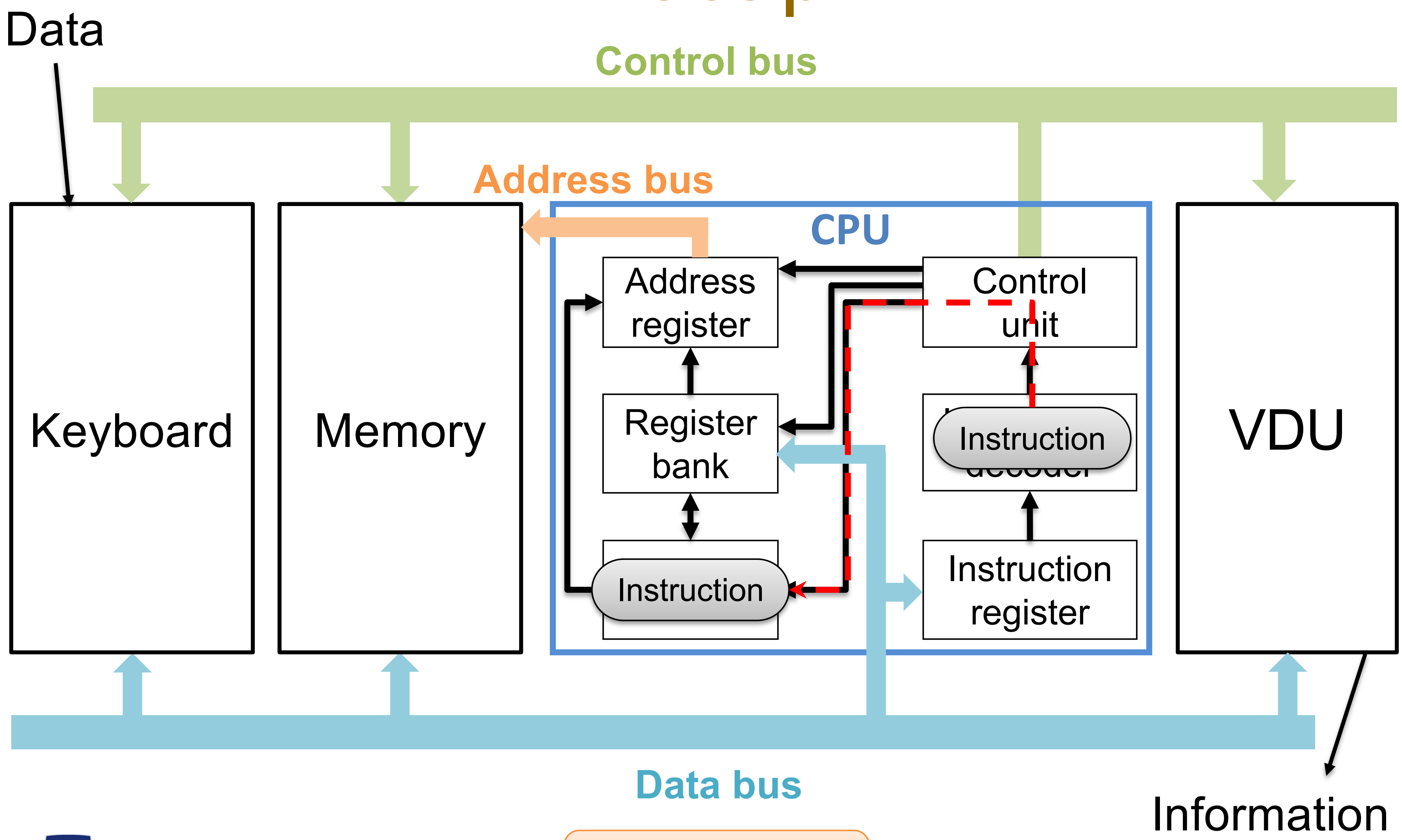
UNIVERSITY OF LIVERPOOL

Recap



Decode

Recap



Execute

Simple instructions (MOVS1)

- One of the simplest instructions is to move a value into a register
  - E.g. move 114 into register r3
- The machine code for this instruction is 0x2372 or 0010 0011 0111 0010
- After the instruction is executed register r3 will hold the value 0x00000072 (hexadecimal for 114) and value in register r15, the program counter, will have increased by 2.
- All other registers remain unchanged.

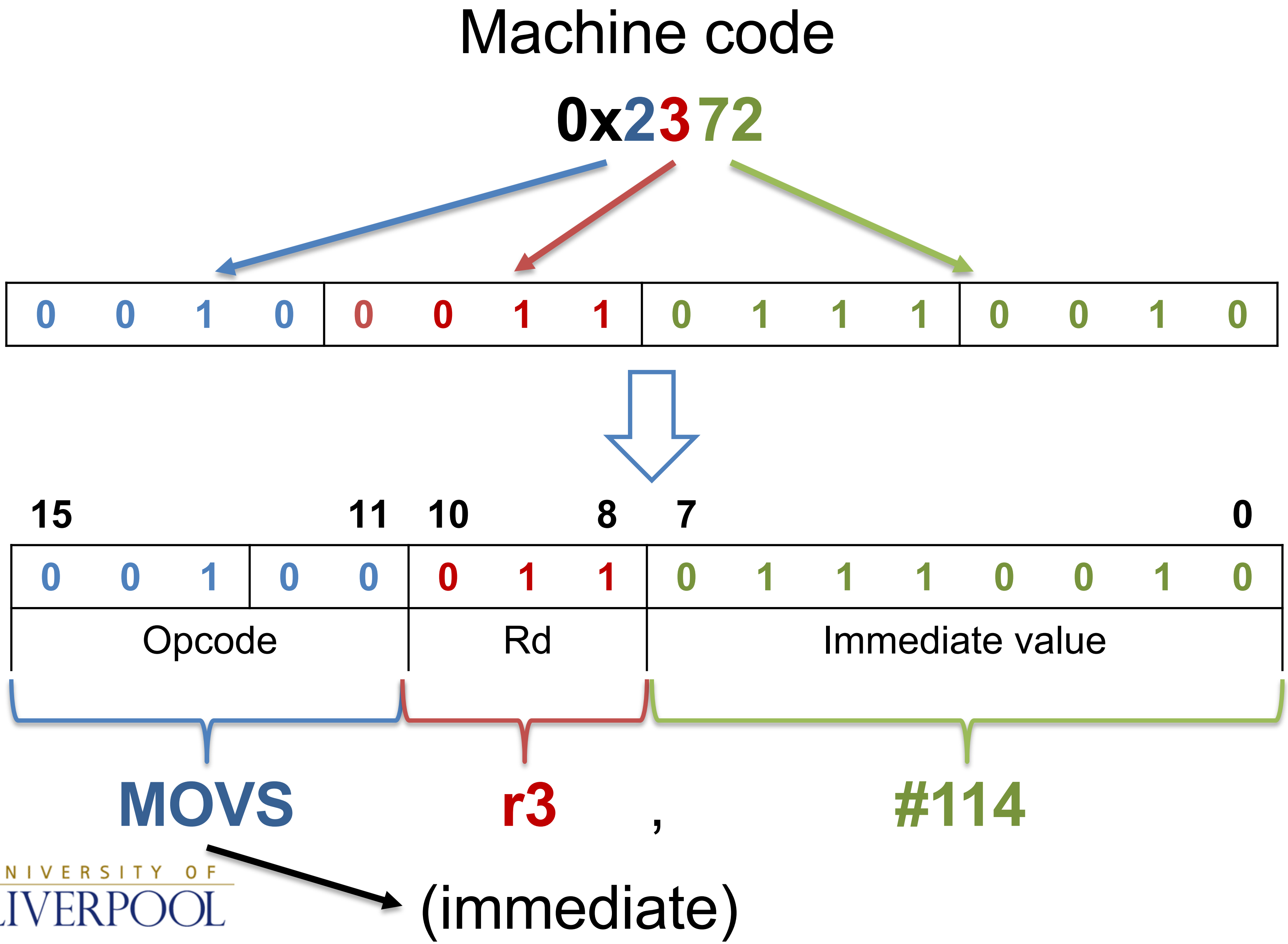
Simple instructions (MOVS1)

- MOVS r3, #114

15				11	10		8	7							0
0	0	1	0	0	0	1	1	0	1	1	1	0	0	1	0
Opcode					Rd			Immediate value							

- Opcode (type of operation) = b00100
  - Move operation - it tells the control unit and decode cycle (ID) what type of instruction it is
- Rd (destination register) = r3
- Immediate value = $01110010_2 = 114_{10} = 0x72$

Simple instructions (MOVS1)



Machine code (MOVS1)

- Look at the machine code for the instruction; move 114 into register r3, again.

0x2372

- The value **114** is given in the least significant byte

0x23**72**

114 in decimal is 72 in hexadecimal.

- Register r**3** is given by the 2nd digit

0x2**3**72

Question

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

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

Registers r1, r3 and r15 hold the values 0xCCDDEEFF, 0xFEDCBA98 and 0x00000108 respectively. What is their value after the execution of the instruction 0x21CB?

- r1=0x000000AB, r3=0xFEDCBA98, r15=0x0000010C
- r1=0x000000A0, r3=0xFEDCBA9C, r15=0x0000010B
- r1=0x000000A0, r3=0xFEDCBA9C, r15=0x0000010A
- r1=0x000000CB, r3=0xFEDCBA98, r15=0x0000010A

Total Results: 0

Simple instructions (MOVS2)

- Another simple instruction is to move the value in one register to another register
  - E.g. move into r6 the value held in r5 (‘copy’)
- Machine code for this instruction is 0x002E or 0000000000101110
- After the instruction is executed register r6 will hold the same value as held in register r5
- and value in register r15, the program counter, will have increased by 2.
- Other registers including r5 are unchanged.

Simple instructions (MOVS2)

- MOVS r6, r5

15		13	12	11	10				6	5		3	2		0
0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0
		Opcode									Rm			Rd	

  - Rm (source register) = r5 = b101
  - Rd (destination register) = r6 = b110

Simple instructions (MOVS2)

Machine code

0x002E

0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0

15				11	10				6	5		3	2		0
0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0
Opcode										Rm			Rd		

MOVS **r5** , **r6**

(register)

Machine code (MOVS2)

- Look at the machine code for the instruction; move into register r**6** the value in register r**5**

0000 0000 0010 1110

- Register r**6** is given by the 3 least significant bits

0000 0000 0010 1**110**

- Register r**5** is given by the next 3 bits

0000 0000 00**10 1**110

0 0 2 E_{16}

conversion table

Question

Registers r0, r7 and r15 hold the values 0xCCDDEEFF, 0xFEDCBA98 and 0x0000010A respectively. What is their value after the execution of the instruction 0x0038?

- r0=0xFEDCBA9C, r7=0xFEDCBA9C, r15=0x0000010A
- r0=0xFEDCBA98, r7=0xFEDCBA98, r15=0x0000010C
- r0=0xFEDCBA88, r7=0xFEDCBA88, r15=0x0000010A
- r0=0xFEDCBA9C, r7=0xFEDCBA98, r15=0x0000010A

Total Results: 0

Mnemonics

In general nobody remembers all of the machine code for any particular processor (or indeed any).

Instead we use mnemonics - mnemonics are words or phrases which are easy to remember and can replace something which is difficult to remember.

The instruction to move the value 114 into register r3 has the mnemonic “MOVS r3, #114”.

This is much easier to remember than 0x2372

Note the “#” means it is an “immediate” value

Mnemonics

Every instruction has both a machine code and a mnemonic.

Consider the instructions from the last and today lectures.

Machine Code		Mnemonic
0x2372	⏩	MOVS r3, #114
0x21CB	⏩	MOVS r1, #0xCB
0x002E	⏩	MOVS r6, r5
0x0038	⏩	MOVS r0, r7

UNIVERSITY OF LIVERPOOL

Assembly Language

Assembly (ASM) is a low level programming language, which is specific to the architecture (processor).

A list of mnemonics for every instruction that it is execute sequentially is an assembly language program.

E.g.

```
MOVS r6, r5
MOVS r1, #0xCB
MOVS r0, r7
MOVS r3, #114
```

Assembler

A computer package called an assembler converts an assembly language program into a machine code program.

Mnemonic		Machine Code
MOVS r3, #114	⏩	0x2372
MOVS r1, #0xCB	⏩	0x21CB
MOVS r6, r5	⏩	0x002E
MOVS r0, r7	⏩	0x0038

Memory	Address
0x23	0x00007000
0x72	
0x21	0x00007002
0xCB	
0x00	0x00007004
0x2E	
0x00	0x00007006
0x38	

The machine code can be downloaded to the microprocessor memory; each instruction occupying 2 adjacent memory locations.

Compiler

A high level language (such as Python, Java, C, C++, Fortran, Pascal, etc.) is converted into either machine code or mnemonics using a computer package called a compiler.

Most applications are written in a high level language but assembly language programming is commonly used for engineering systems which must operate in real time e.g. a mobile phone.

Nobody writes computer programs using machine code - it is too slow and error prone.

Instructions for Arithmetic

The ARM Cortex M0 can add, subtract and multiply numbers (but not divide).

The mnemonic for add the value in register x to the value in register y and place the sum in register z is:

ADDS rz, ry, rx $\rightarrow$ rz = ry + rx

- E.g. to add the value in register r1 to the value in register r2 and leave the sum in register r3 the mnemonic is:

  ADDS r3, r2, r1

We don't need to know the machine code, the assembler will generate it for us.

Subtraction

Similarly the mnemonic for subtract the value in rx from the value in ry and place the difference in rz is:

SUBS rz, ry, rx $\rightarrow$ rz = ry - rx

Note that the order is important and if the value in rx is greater than the value in ry then a negative result will be placed in rz.

Negative numbers are expressed in the two's complement format.

Question

When poll is active, respond at **PollEv.com/elec211**
Text **ELEC211** to **22333** once to join

r1, r2 and r15 hold the values 0x00000010, 0x0000000A and 0x0000010C respectively.
What values are held in r1, r2, r3, r4, r5 and r15 after the execution of the following?

ADDS r3,r2,r1
ADDS r4,r1,r2
SUBS r5,r1,r2
(16 bit instructions)

- r1=0x00000010, r2=0x0000000C, r3=0x00000016, r4=0x00000016, r5=0x00000006, r15=0x00000112
- r1=0x00000010, r2=0x0000000A, r3=0x0000001A, r4=0x0000001A, r5=0x00000006, r15=0x00000112
- r1=0x00000012, r2=0x0000000A, r3=0x0000001A, r4=0x00000016, r5=0x00000006, r15=0x0000010C
- r1=0x00000010, r2=0x0000000C, r3=0x00000016, r4=0x0000001A, r5=0x00000006, r15=0x0000010C

Total Results: 0

Reverse Subtraction

There is also a ‘reverse subtraction’ instruction;

that is the same as subtraction but the register ry is subtracted from zero. The mnemonic is:

RSBS rz, ry, #0 $\rightarrow$ rz = 0 - ry

The meaning of the instruction is ‘subtract the value in ry from zero and place the difference in rz’.

Multiplication

The mnemonic for multiply the value in rx by the value in ry and place the product in rx is:

MULS rx, ry, rx $\rightarrow$ rx = ry $\times$ rx

If the result is more than 32 bits long the destination register, rx, only holds the bottom 32 bits of the result and the rest is lost.

Overflow

In the ARM CPU register bank r0-r15 are 32 bits wide.

Any operation that generates a result wider than 32 bits is producing an overflow.

- E.g. the product in decimal of 100,000 and 50,000 is 5,000,000,000.
- In hexadecimal the result of the product of 0x186A0 and 0xC350 is 0x12A05F200 which in binary is:

  $1\ 0010\ 1010\ 0000\ 0101\ 1111\ 0010\ 0000\ 0000_2$

- The result is 33 bits long.
- The destination register holds the bottom 32 bits (0x2A05F200) and the most significant bit is lost.

The same can happen with addition.

UNIVERSITY OF LIVERPOOL

Question

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

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

r1, r2, r3, r7 and r15 hold the values 0x00001020, 0x00000005, 0xABC00000, 0x00000010 and 0x00000112 respectively.

What values are held in r1, r2, r3, r7 and r15 after the execution of the following?

MULS r1,r2,r1

MULS r3,r7,r3

(16 bits instructions)

- r1=0x000050A0, r2=0x00000005, r3=0xBC000000, r7=0x00000010, r15=0x00000116
- r1=0x000050A0, r2=0x00000005, r3=0xABC00000, r7=0x00000010, r15=0x00000116
- r1=0x000050A8, r2=0x00000015, r3=0xBC000000, r7=0x00000010, r15=0x00000118
- r1=0x000050A0, r2=0x00000005, r3=0xABC000000, r7=0x0000001A, r15=0x00000118

Total Results: 0

Summary

- Simple instructions
- Mnemonics
- Assembly Language
- Types of instructions (Arithmetic)

Next class?

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