Digital Electronics and Microprocessor Systems (ELEC211)

Dave McIntosh and **Valerio Selis**

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

UNIVERSITY OF LIVERPOOL

Outline

Types of instructions
  - Logical Instructions
- Addressing modes
  - Register addressing
  - Immediate addressing
  - Indirect addressing
- Limitations and restrictions
- Load/Store architecture
  - Load instructions
  - Store instructions

Instructions using logic

As well as arithmetic the ARM CPU can also do logic such as AND and OR.

AND value in rx with value in ry and leave the result in ry has the mnemonic:

ANDS ry, rx → ry = ry AND rx

OR value in rx with value in ry and leave the result in ry has the mnemonic:

ORRS ry, rx → ry = ry OR rx

Logical AND

Logical functions are performed bit by bit so if we AND 0xFEDCBA98 with 0x11223344 :

```
      1111 1110 1101 1100 1011 1010 1001 1000
AND   0001 0001 0010 0010 0011 0011 0100 0100
=     0001 0000 0000 0000 0011 0010 0000 0000
```

The result is 0x10003200.

UNIVERSITY OF LIVERPOOL

conversion table

Logical OR

Similarly if we OR 0xFEDCBA98 with 0x11223344

```
      1111 1110 1101 1100 1011 1010 1001 1000
ORR   0001 0001 0010 0010 0011 0011 0100 0100
=     1111 1111 1111 1110 1011 1011 1101 1100
```

The result is 0xFFFEBBDC.

UNIVERSITY OF LIVERPOOL

conversion table

Exclusive OR (XOR)

Exclusive OR has the mnemonic EORS ry, rx
To exclusive OR 0xFEDCBA98 with 0x11223344:

```
      1111 1110 1101 1100 1011 1010 1001 1000
EOR   0001 0001 0010 0010 0011 0011 0100 0100
=     1110 1111 1111 1110 1000 1001 1101 1100
```

The result is 0xEFFE89DC.

UNIVERSITY OF LIVERPOOL

conversion table

Bit clear

‘Bit clear’ performs the function ‘ry AND NOT(rx)’ and has mnemonic BICS ry, rx
A logic 1 in rx ‘clears’ a 1 bit in ry.
To bit clear 0xFEDCBA98 with 0x11223344:

	1111 1110 1101 1100 1011 1010 1001 1000
BIC	0001 0001 0010 0010 0011 0011 0100 0100
=	1110 1110 1101 1100 1000 1000 1001 1000

The result is 0xEEDC8898.

UNIVERSITY OF LIVERPOOL

conversion table

Question

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

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

r1 holds the value 0x00000101 and r2, r3, r4, r5, r6 all hold 0x00000011.
What values are held in r1, r2, r3, r4, r5 and r6 after executing the instructions?
ANDS r2,r1
ORRS r3,r1
EORS r4,r1
BICS r5,r1
BICS r1,r6
(16 bits instructions)

- r1=0x00000101, r2=0x00000011, r3=0x00000111, r4=0x00000110, r5=0x00000011
- r1=0x00000100, r2=0x00000001, r3=0x00000111, r4=0x00000110, r5=0x00000010
- r1=0x00000100, r2=0x00000001, r3=0x00000011, r4=0x00000110, r5=0x00000011
- r1=0x00000100, r2=0x00000011, r3=0x00000111, r4=0x00000110, r5=0x00000010

Total Results: 0

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

UNIVERSITY OF LIVERPOOL

Addressing modes

The concept of addressing mode is common to all microprocessors and an understanding of addressing mode is important for users of microprocessors.

The ARM Cortex M0 microprocessor supports many different addressing modes and we will concentrate on just four.

When accessing an operand for data processing there are several techniques (addressing modes) to specify the desired location.

Register Addressing

Register addressing means that the machine code includes a number (or numbers) that identifies a register (or registers).

E.g. 0000 0000 0010 1110, which is binary code for MOVS r6, r5 - move into r6 the contents of r5.

All of the instructions we have already covered use register addressing.

E.g. SUBS r5, r1, r2
MULS r3, r7, r3
BICS r1, r6

Restricted Register Use

Not all instructions can access all registers.

The instruction with mnemonic MOVS, e.g.

MOVS ry, rx is restricted to the ‘low’ registers, that is registers in the range r0 to r7 inclusive.

A different instruction with mnemonic:

MOV ry, rx allows all 16 registers (including the ‘low’ registers) to be used

e.g. MOV r13, r10 has machine code 0x46D5 or 0100 0110 1101 0101

Immediate Addressing

Immediate addressing means that the machine code contains a value to be used.

E.g. 0x21CB which is the code for

MOVS r1, #0xCB - move into r1 the value 0x000000CB - the # means ‘immediate’.

This instruction uses both register addressing for r1 and immediate addressing for 0xCB ($=203_{10}$)

More Immediate Addressing

Immediate addressing can be used to replace the last register in addition & subtraction instructions.

E.g. ADDS r2, r1, #5

- add 5 to value in r1 and put the sum in r2

Machine code is 0x1D4A or 0001 1101 0100 1010

SUBS r6, r6, #99 - subtract 99 from value in r6

Machine code is 0x3E63 or 0011 1110 0110 0011

Decimal 99 is equal to hexadecimal 0x63

Question

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

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

What values are held in r4, r6 and r7 after the execution of the following?

MOVS r4,#17

ADDS r7,r4,#4

SUBS r6,r7,#7

ADDS r4,r4,#250

- r4=0x0000010B, r6=0x0000000E, r7=0x00000015
- r4=0x0000010B, r6=0x0000001E, r7=0x00000015
- r4=0x00000013, r6=0x0000000E, r7=0x00000013
- r4=0x00000011, r6=0x00000010, r7=0x00000015

Total Results: 0

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

UNIVERSITY OF LIVERPOOL

Limitation of immediate addressing

There is one problem with immediate addressing when using a processor like the ARM.

The instruction code is 16 bits (sometimes 32 bits) and it must include information about the type of instruction (e.g. ADDS, MOVS, etc.) and the destination register as well as the immediate value.

So a 32 bit value can not be put into a 32 bit register using immediate addressing and MOVS.

E.g. the following instruction is not allowed

```
MOVS r1, #0xF97D5EC5
```

Restrictions – 8 bits

Immediate values are restricted to a given number of bits e.g. 3, 5, 7 or 8 bits.

8 bit values are allowed for the MOVS instruction and for ADDS/SUBS instructions where the destination register and the source register are the same e.g.

ADDS r6, r6, #99 - add 99 from value in r6
Machine code is 0x3663 or 0011 0110 0110 0011

Decimal 0 to 255 (inclusive) is the allowed range for 8 bit values.

Restrictions – 3 bits

3 bit immediate values are allowed for ADDS and SUBS instructions where the destination register is different from the source register.

E.g. SUBS r3, r1, #5

- subtract 5 from value in r1 and put the difference in r3

Machine code is 0x1F4B or 0001 1111 0100 1011

Decimal 0 to 7 (inclusive) is the allowed range for 3 bit values.

Question

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

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

Which of the following instructions uses an immediate value that is allowed in the ARM Cortex M0?

(A) MOVS r7,#0x10B; 267_{10}

(B) ADDS r3,r3,#0x0A4; 164_{10}

(C) SUBS r0,r6,#0x03; 3_{10}

(D) ADDS r5,r2,#0x0A4; 164_{10}

- (A)=Allowed, (B)=Allowed, (C)=NotAllowed, (D)=Allowed
- (A)=NotAllowed, (B)=NotAllowed, (C)=Allowed, (D)=NotAllowed
- (A)=NotAllowed, (B)=Allowed, (C)=Allowed, (D)=NotAllowed
- (A)=Allowed, (B)=NotAllowed, (C)=NotAllowed, (D)=Allowed

Total Results: 0

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

UNIVERSITY OF LIVERPOOL

Indirect Addressing

Indirect (or register indirect) addressing uses a value in a register to identify a memory address
e.g. the instruction:

LDR rd, [rn]

- rd is the register to be loaded
- [rn] denotes that we are using the value contained in register rn as a memory address

means put (or 'load') into register rd the data held in the memory location that has the address given by the value in register rn.

Indirect Addressing - Example

So if r1 holds the value 0x00004000 and the memory location with address 0x00004000 holds the value 0xF97D5EC5 then:

LDR r3, [r1]

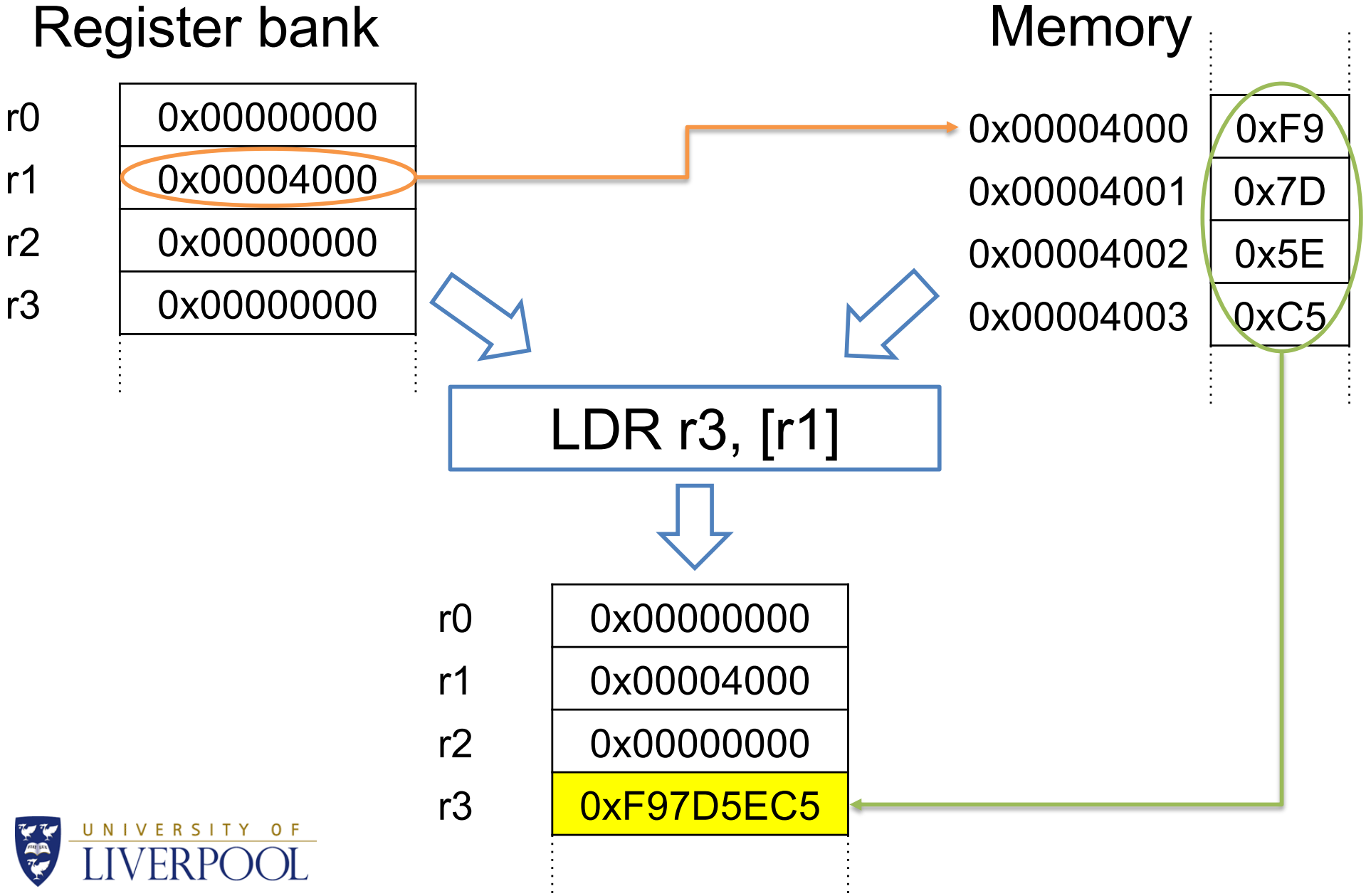
would load 0xF97D5EC5 into r3. Whereas

MOV r3, r1

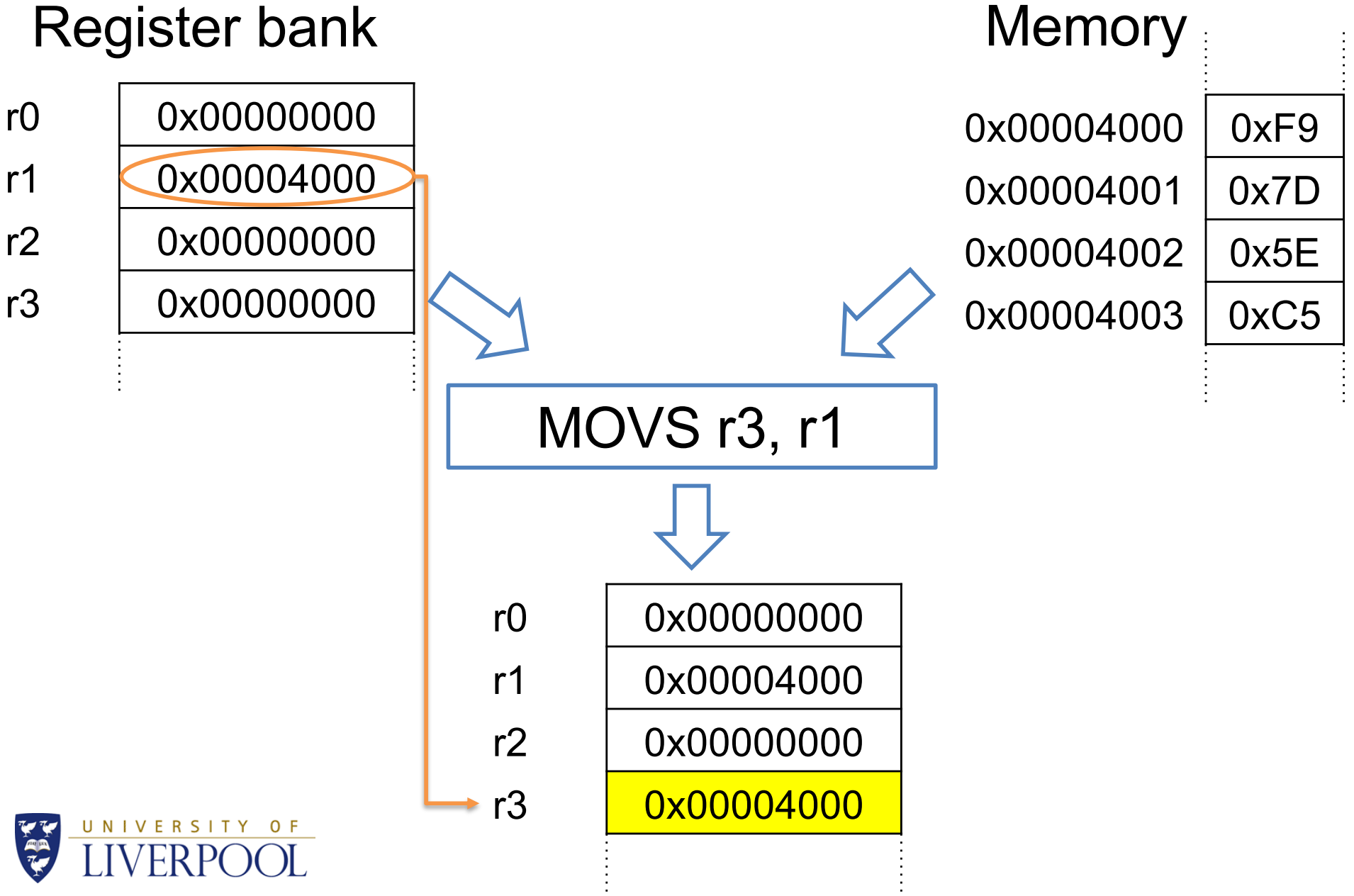
would load 0x00004000 into r3.

Note that the square brackets [ ] denote indirect addressing.

Indirect Addressing - Example



Indirect Addressing - Example



From register to memory

The load instruction, unlike MOV, can be used to put a true 32 bit value into a register.

Another important instruction that uses indirect addressing is ‘store’ which we can think of as being the opposite of ‘load’. The instruction

STR rd, [rn]

- rd is the register whose contents are to be stored
- [rn] denotes that we are using the value contained in register rn as a memory address

means put (or ‘store’) the data from register rd into the memory location that has the address given by the value in rn.

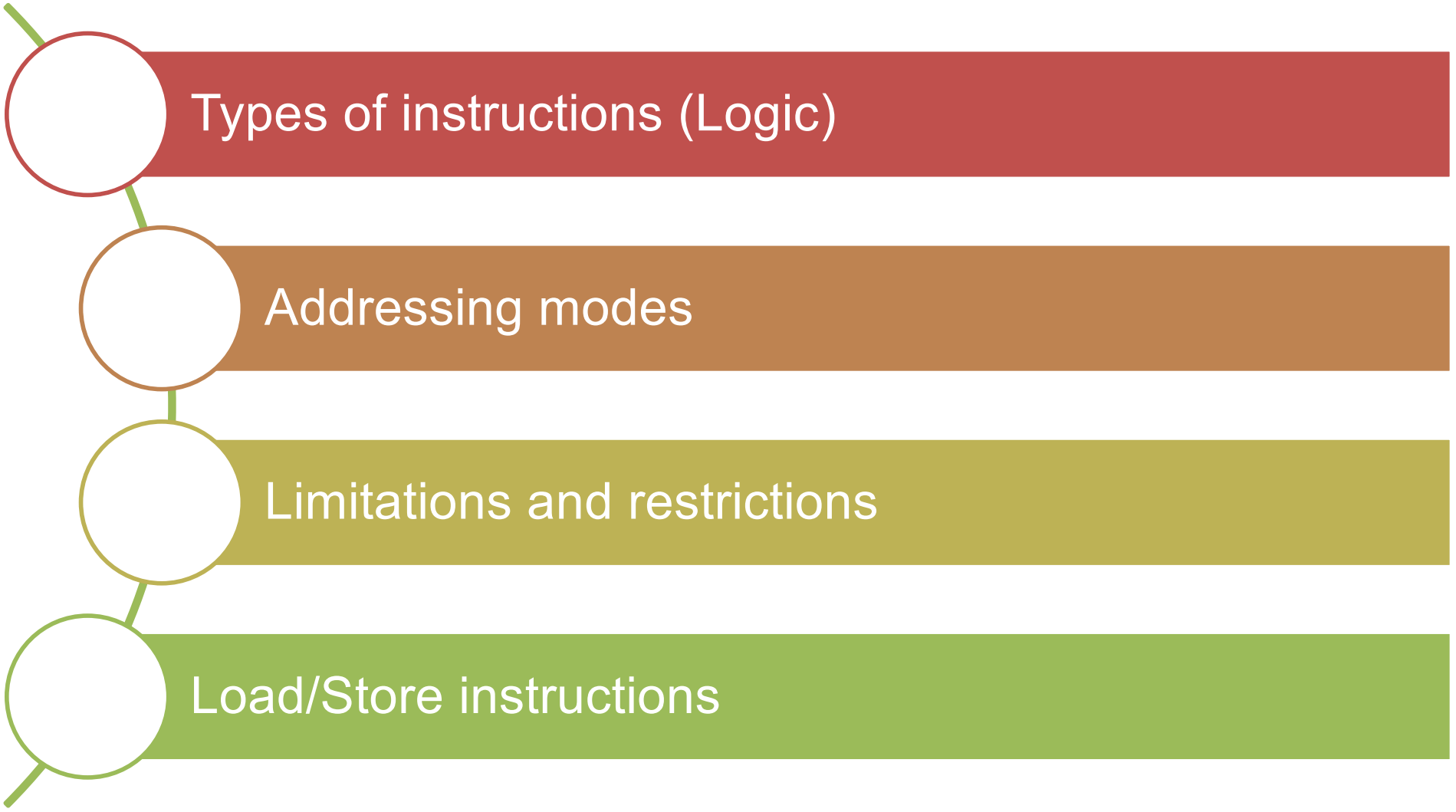
Load and Store

When a LOAD instruction is executed the data travels from memory to register

whereas when a STORE instruction is executed the data travels from register to memory.

LOAD is a memory ‘read’ and STORE is a memory ‘write’.

Summary


Types of instructions (Logic)
Addressing modes
Limitations and restrictions
Load/Store instructions

Next class?

Tomorrow at 2 p.m. in the
Building 502,
Lecture Theatre 2
(502-LT2)