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
- The barrel shifter
  - 8x8 bit cross bar barrel shifter
- Shift instructions
  - LSLS, LSRS, ASRS, RORS
- Shift by an immediate
- Shift by a register
- Multiplying and dividing by 2

Recap – flags

Flags are stored in the Current Program Status Register (CPSR):

- Indicate what happened as a consequence of executing instructions
- Controlled by using the item {S} in the instruction

The zero flag (Z) is set when the **result** (not including any carry out) is 0x00000000.

- Used for programs with “loops”
- E.g.:

```
MOVS r0, #0
SUBS r1, r2, r2
```

UNIVERSITY OF LIVERPOOL

Recap – flags

The negative flag (**N**), is set when the most significant bit of the 32 bit **result** is 1.

- In hexadecimal when the most significant digit is between 8 and F.
- E.g.:

  1100 0001 1001 0000 1010 1011 0000 0000

  Negative number ➔ negative flag set

  0x4190AB00

  Positive number ➔ negative flag cleared

- To check if a value in a register is negative or positive we could use:

  MOV**S** rn, rn

Recap – flags

The carry flag (**C**) is set when the **result** is greater than $(2^{32}–1)$.

- Used to check if the result of an operation with unsigned integer values is correct or not
- Can be used to add add together numbers greater than $(2^{32}-1)$ with the ADCS instruction
- E.g.:

```
  1111 1111 1111 1111 1111 1111 1111 1111
+ 0000 0000 0000 0000 0000 0000 0000 0010
------------------------------------------
1 0000 0000 0000 0000 0000 0000 0000 0001
```

UNIVERSITY OF LIVERPOOL

Recap – flags

The overflow flag (**V**) is set when the **result** is less than -2^{31} or greater than $(2^{31}-1)$

- Result out of the allowed range (0x80000000 to 0x7FFFFFFF)
- Used to check if the result of an operation with two's complement values is correct or not
- This is set when the addition of 2 positive numbers gives in a negative result (overflow in the sign bit)
- E.g.:

```
  0101 1001 0110 1000 0010 1111 0000 0000
+ 0100 0001 1001 0000 1010 1011 0000 0000
  ---------------------------------------
  1001 1010 1111 1000 1101 1010 0000 0000
```

UNIVERSITY OF LIVERPOOL

Recap

Branches:

- Unconditional branch:
  - Always executes and it reloads the program counter with a new value.
- Conditional branch:
  - Depends on the condition (**status of the flags**):
    - It will branch
    - Or it will continue normal execution

UNIVERSITY OF LIVERPOOL

Recap

Signed integer numbers

- Sign & Magnitude method

31	30	0
s	Same number for positive or negative representation	

- 2's complement method
  - A negative number, –x, is given by 2^n – x, where n = 32 (32 bit). The sign bit (**s**) is automatically set

Floating point numbers

- IEEE 754 standard (single precision format)

31	30	23	22	0
s	Exponent (**n**)		Mantissa (**a**)	

The barrel shifter

The barrel shifter is a very useful feature of the ARM Cortex M0 microprocessor which allows bit patterns to be rotated.

E.g. LSLS r1, r2, **#5**

would take the bit pattern in register r2 and shift it **5 places** to the **left** before placing it in register r1.

So if r2 held:

1010 0101 1100 0011 1001 0110 1110 0111

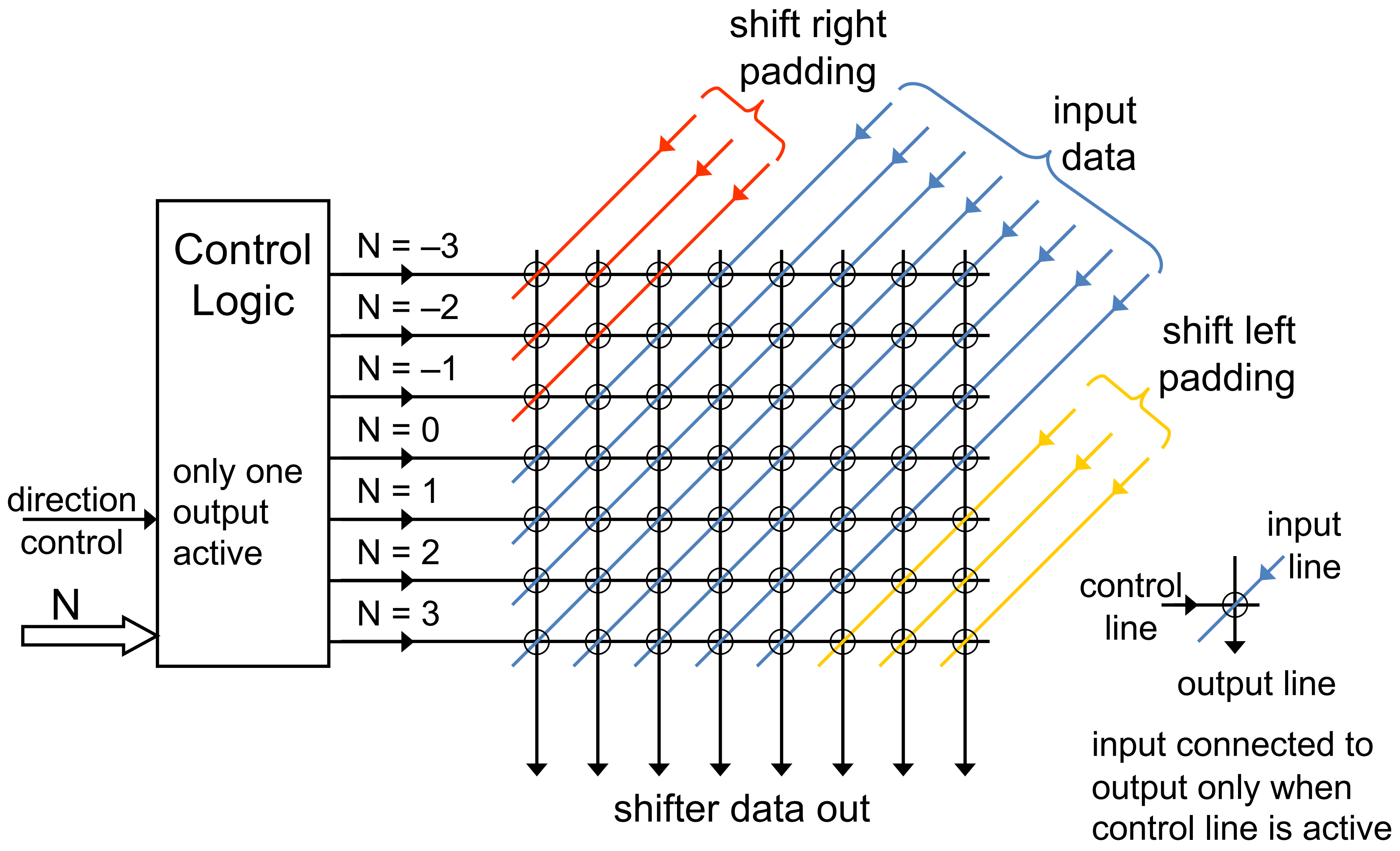
after the instruction is executed r1 would hold:

1011 1000 0111 0010 1101 1100 111**0 0000**

$\xleftarrow{\hspace{10em}}$

UNIVERSITY OF LIVERPOOL

Barrel Shifter (8 bits)



UNIVERSITY OF LIVERPOOL

Different shifts

There are four different shift instructions:

1. LSLS

   Logical shift left - the bits are shifted to the left and the new bits added in at the right hand side are 0

2. LSRS

   Logical shift right - the bits are shifted to the right and the new bits added in at the left hand side are 0

Different shifts

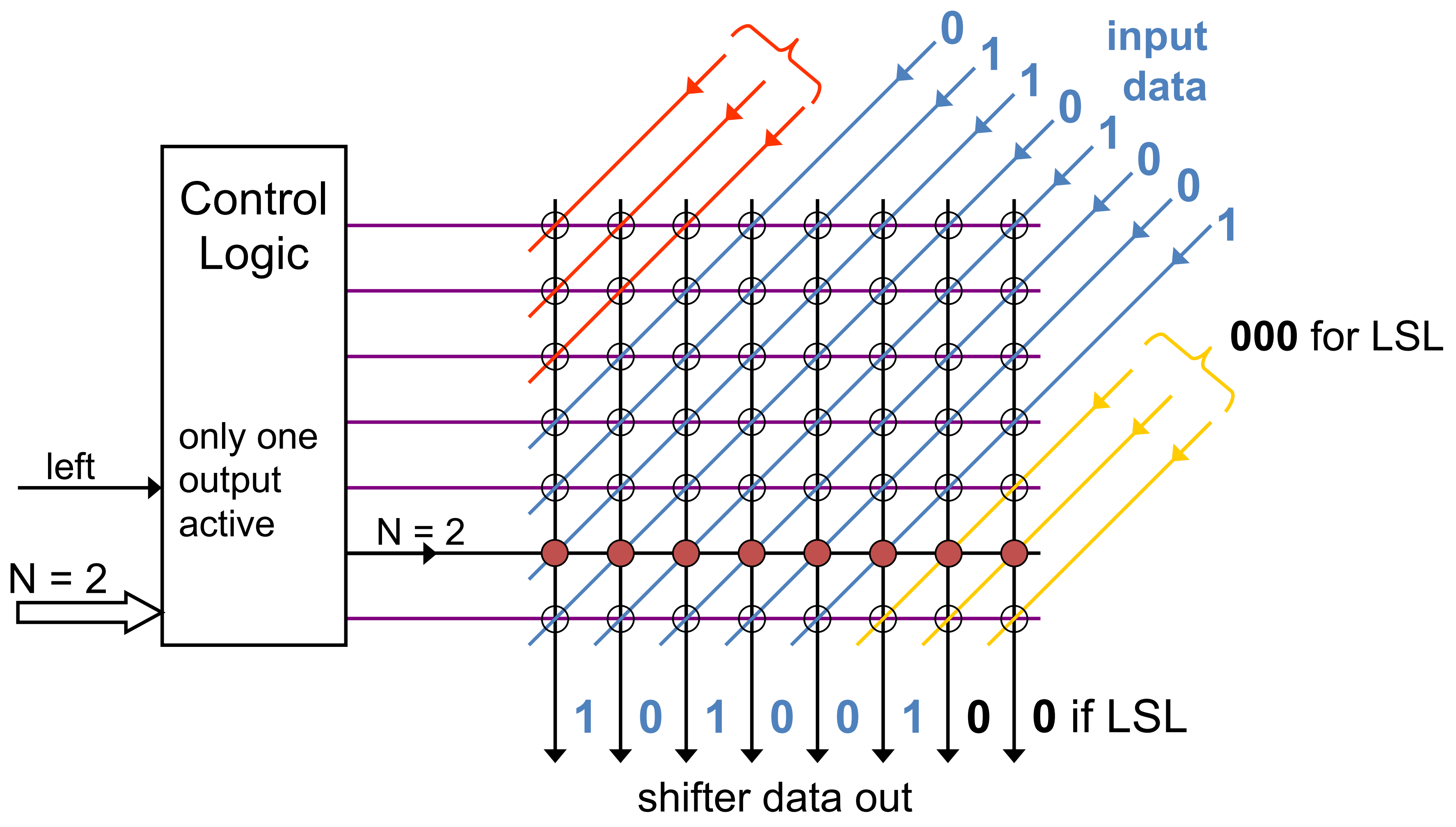
3. ASRS

   Arithmetic shift right - bits are shifted rightwards and the new bits added in are the same as the ‘old’ sign bit

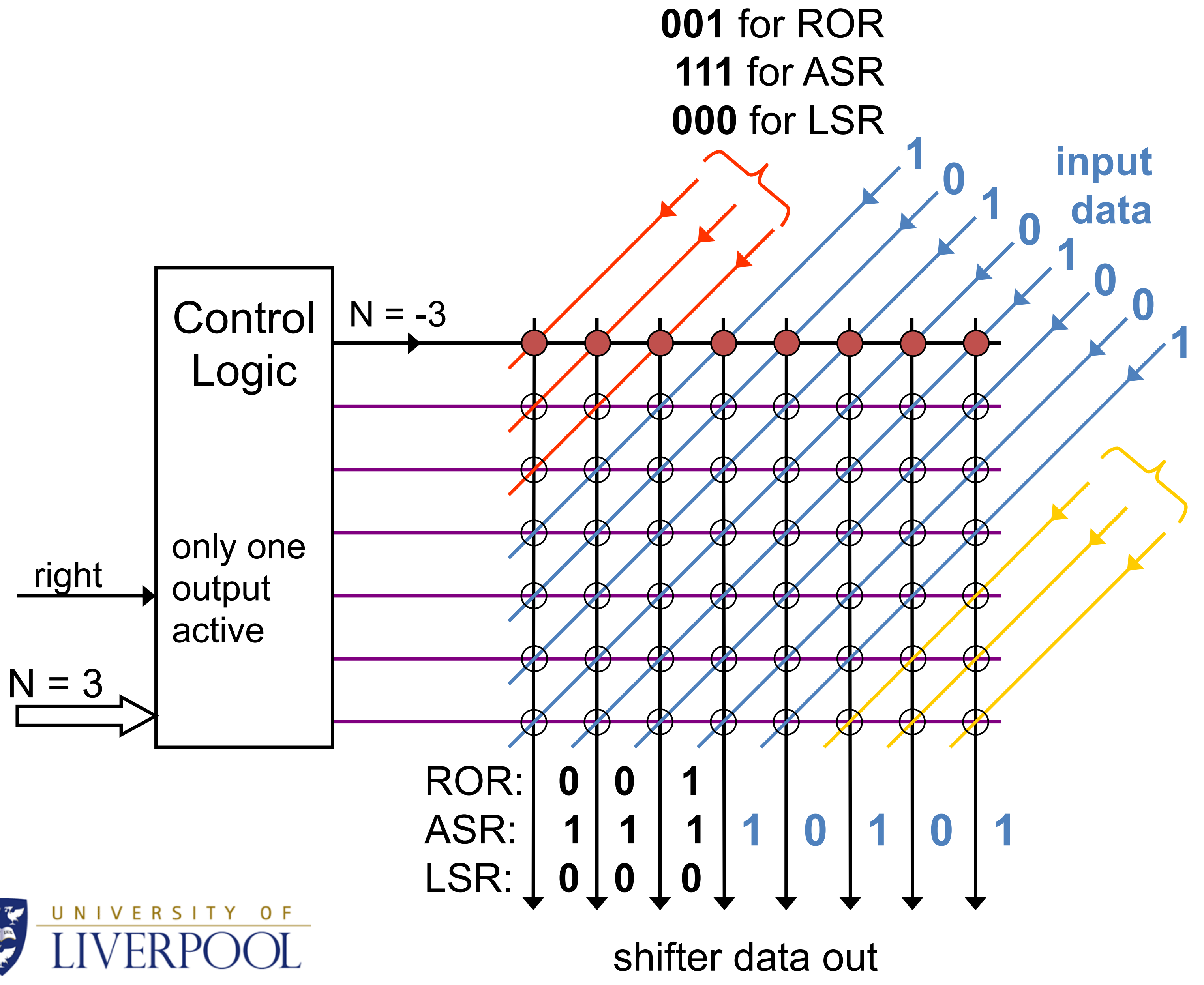
4. RORS

   Rotate right (ROR) - bits are rotated rightwards so that the bits shifted out at the right hand side reappear at the left hand side

Example: Shift left by 2



Example: Shift right by 3



Shift by an immediate

LSLS, LSRS and ASRS (but not RORS) can use an immediate to determine the number of bits required for the shift.

E.g. LSRS rx, ry, #n

Shift the bit pattern in register ry to the **right** by **n bits** and put this value in register rx. The most significant **n bits** of rx are set to 0.

- The registers must be 'low' registers, r0 to r7, and the immediate is limited to **5 bits** ($0 \leq n \leq 31$).

Shift by a register

All shifts including RORS can use the least significant byte of the value in register to determine the number of bits required for the shift.

E.g. RORS ry, ry, rz

Rotate **right** the bits in **ry** by the value in the least significant byte, *n*, of **rz**. The least significant *n* bits of **ry** are copied into the most significant *n* bits of **ry**.

- The source and destination registers must be the same.
- The registers must be 'low' registers, r0 to r7.

UNIVERSITY OF LIVERPOOL

Shift by a register

E.g. RORS r2, r2, r1

So if r1 held:

0x10000005

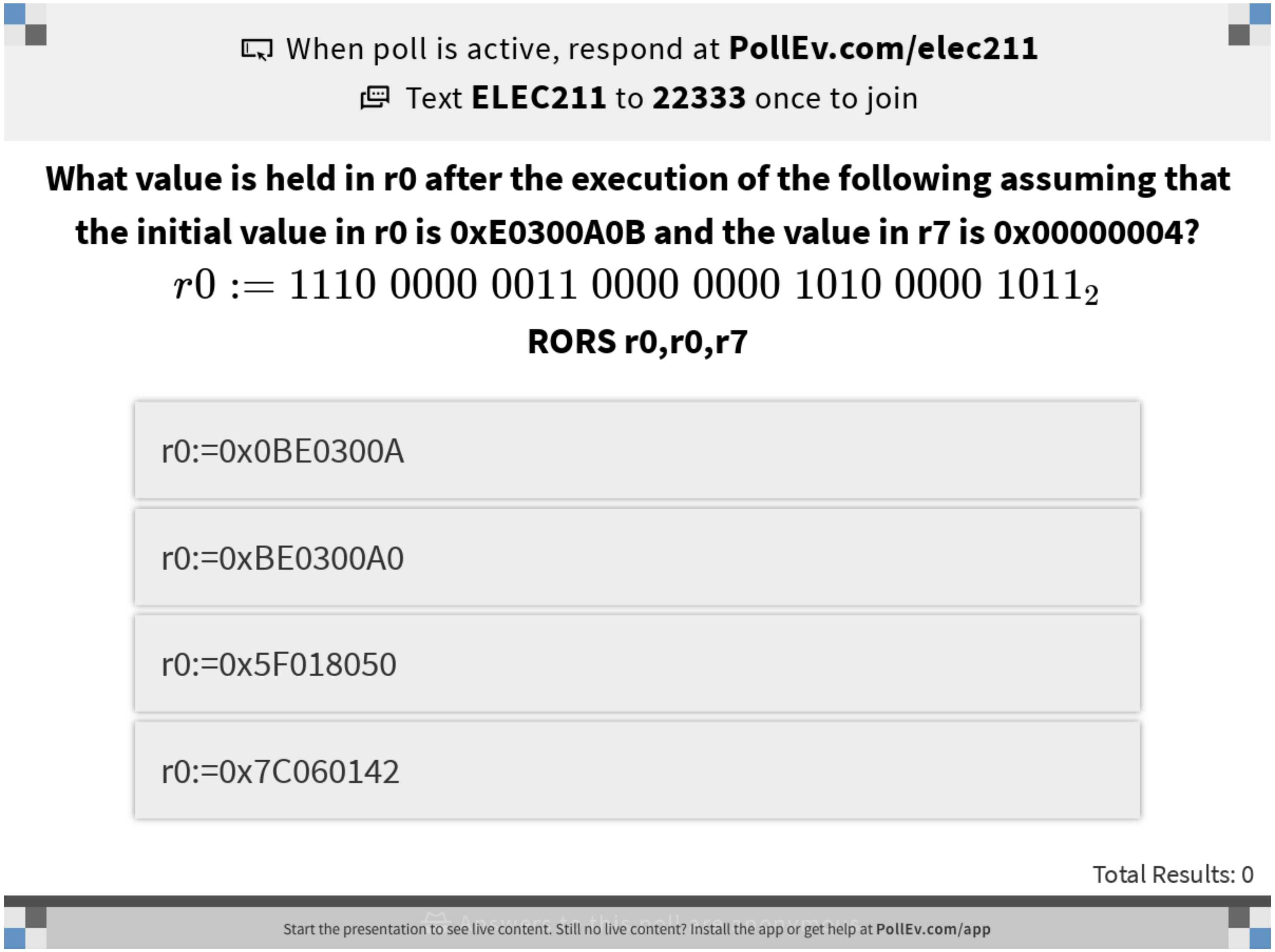
And r2 held:

1010 0101 1100 0011 1001 0110 1110 0111

after the instruction is executed r2 would hold:

0011 1101 0010 1110 0001 1100 1011 0111

Question



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

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

What value is held in r0 after the execution of the following assuming that the initial value in r0 is 0xE0300A0B and the value in r7 is 0x00000004?

$$r0 := 1110\ 0000\ 0011\ 0000\ 0000\ 1010\ 0000\ 1011_2$$

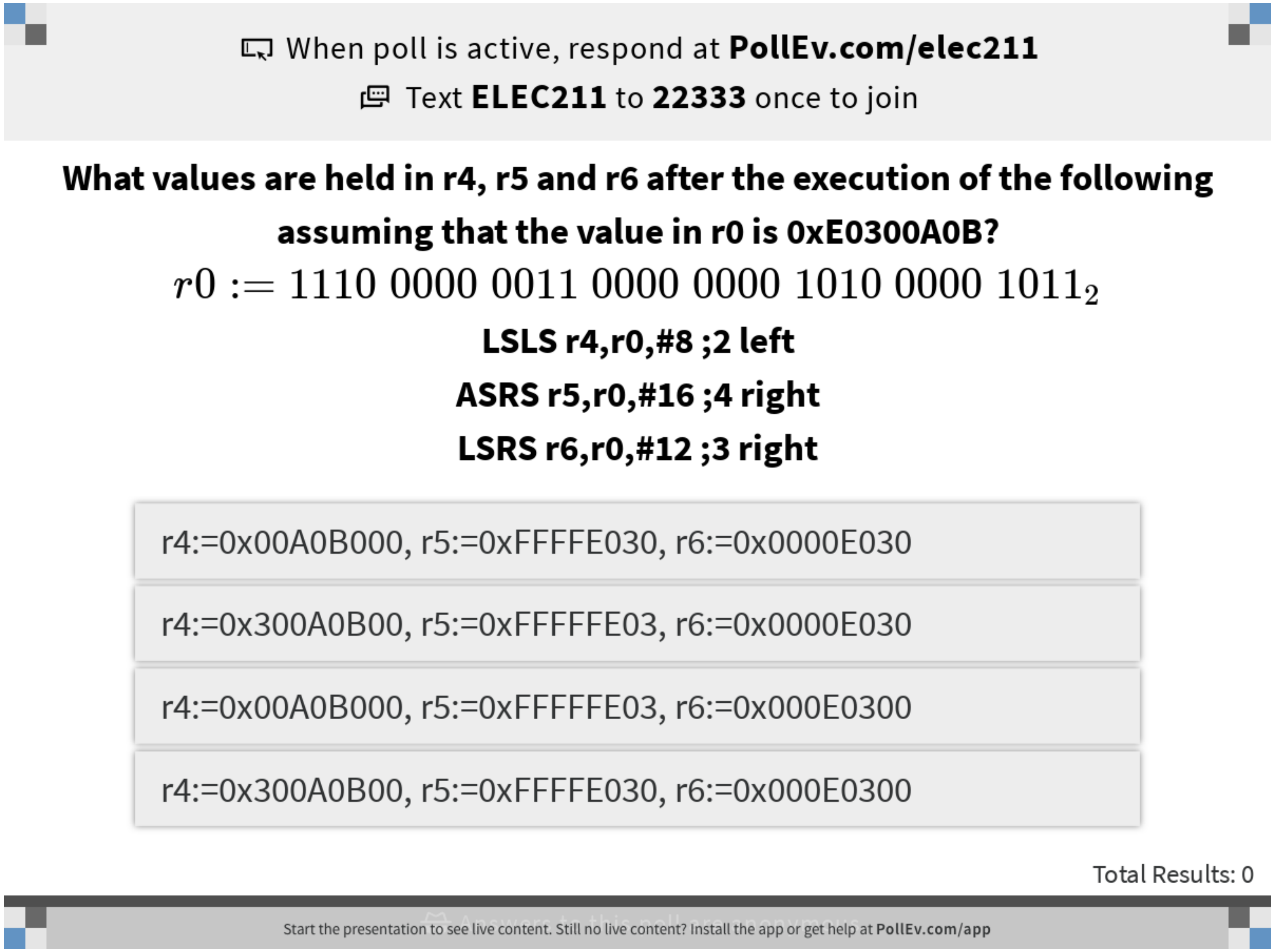
RORS r0,r0,r7

- r0:=0x0BE0300A
- r0:=0xBE0300A0
- r0:=0x5F018050
- r0:=0x7C060142

Total Results: 0

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

Question



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

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

What values are held in r4, r5 and r6 after the execution of the following assuming that the value in r0 is 0xE0300A0B?

$$r0 := 1110\ 0000\ 0011\ 0000\ 0000\ 1010\ 0000\ 1011_2$$

LSLS r4,r0,#8 ;2 left

ASRS r5,r0,#16 ;4 right

LSRS r6,r0,#12 ;3 right

- r4:=0x00A0B000, r5:=0xFFFFE030, r6:=0x0000E030
- r4:=0x300A0B00, r5:=0xFFFFFE03, r6:=0x0000E030
- r4:=0x00A0B000, r5:=0xFFFFFE03, r6:=0x000E0300
- r4:=0x300A0B00, r5:=0xFFFFE030, r6:=0x000E0300

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

Using shift to multiply

Logical shift left can be used to multiply by 2^n

- A left shift by 1 is a multiplication by two.

For example:

- LSLS rx, ry, **#n** ; multiplies value in ry by 2^{n}

If we want to multiply by 32 (**n**=$\log_2$ 32=**5**):

- LSLS rx, ry, **#5** ; multiply value in ry by $2^{5} = 32$

If we want to multiply by 1024 (**n**=$\log_2$ 1024=**10**):

- LSLS rx, ry, **#10** ; multiply value in ry by $2^{10} = 1024$

UNIVERSITY OF LIVERPOOL

Using shift to do integer division

Arithmetic shift right can be used to do an integer division by 2^n

For example:

- ASRS rx, ry, #n; divides value in ry by 2^n

For example, if **n=3**, we are dividing by 8 (=2^3):

ry := 0000 0000 0000 0000 0000 0011 1110 1000 (= $1{,}000_{10}$)

rx := 0000 0000 0000 0000 0000 0000 0111 1101 (= 125_{10})

$\xrightarrow{\hspace{10cm}}$

This also works in two's complement

ry := 1111 1111 1111 1111 1111 1100 0001 1000 (= $-1{,}000_{10}$)

rx := 1111 1111 1111 1111 1111 1111 1000 0011 (= -125_{10})

$\xrightarrow{\hspace{10cm}}$

Question

What values are held in r3 and r4 after the execution of the following assuming that the value in r0 is 0x00000005 (= 5_{10})?

```
LSLS r3, r0, #1
LSLS r4, r0, #6
```

Answer

LSLS **r3**, **r0**, **#1**

r0 held 0x00000005 which is:

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0101_2$

shift left by 1 bit so that the value in **r3** is 0x0000000A

$\underleftarrow{0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1010}_2$

$= 10_{10} = (2 \times 5)$, where $2 = 2^1$

LSLS **r4**, **r0**, **#6**

shift left by 6 bits so that the value in **r4** is 0x00000140

$\underleftarrow{0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 01}00\ 0000_2$

$= 320_{10} = (64 \times 5)$, where $64 = 2^6$

UNIVERSITY OF LIVERPOOL

Question

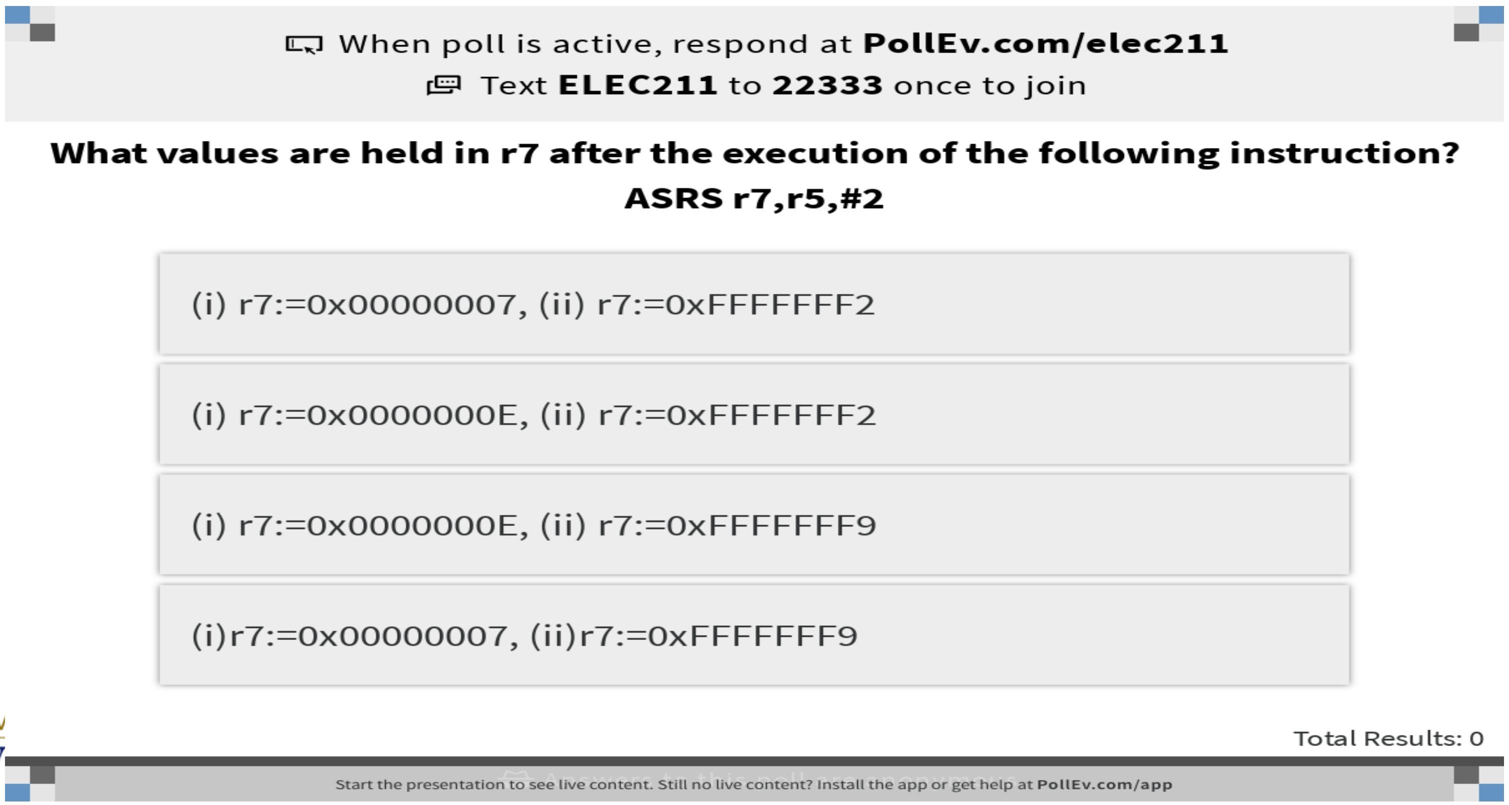
Assuming that the values in r5 are:

(i) 0x0000001C (= 28_{10})

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1100_2$

and (ii) 0xFFFFFFE4 (= -28_{10})

$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110\ 0100_2$



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

What values are held in r7 after the execution of the following instruction?
ASRS r7,r5,#2

- (i) r7:=0x00000007, (ii) r7:=0xFFFFFFF2
- (i) r7:=0x0000000E, (ii) r7:=0xFFFFFFF2
- (i) r7:=0x0000000E, (ii) r7:=0xFFFFFFF9
- (i)r7:=0x00000007, (ii)r7:=0xFFFFFFF9

Total Results: 0

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

Recommended reading

ARM system-on-chip architecture

- Section 4.4: Pages 92-93, the barrel shifter
- Section 3.1: Page 53, Shifted register operands

ARM Assembly Language – an Introduction

- Section 9.3: <shifter operand>

Summary

- Recap
- Barrel shifter
- Shift instructions
- Shift by an immediate and by a register
- Multiplying and dividing by 2

UNIVERSITY OF LIVERPOOL

Next class?

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

UNIVERSITY OF LIVERPOOL