Computer Architecture

Exercise 7

(shifting and masking)

Peter Stallinga

Universidade do Algarve 2024-2025

Shifting is a way to shift a bit pattern to the left or right.

On one side a bit is pushed out of the pattern, on the other side an open space is created. This space is filled with a 0, in so-called *logic* shifts. Note that for unsigned numbers a shift left is a multiplication by 2 (the base number), while a shift right is a division by 2.

```
105 = 01101001
210 = 11010010

100 = 01100100
050 = 00110010
```

A shift by *n* places is a multiplication or division by 2^n. For negative (2's-complement) number divisions this does not work. (Try it). For that to work the *arithmetic* right shift can be used. This keeps the MSB of the bit pattern.

```
-28 = 11100100
-14 = 11110010
```

Masking is a way to read, set, reset and invert bits from a bit pattern.

To **read** a bit from a bit pattern, we mask it with a pattern with a 1 only at the place of the bit that interests us and then compare the result to zero. Imagine we want to know the value of the third least-significant bit:

```
00110101 = A
00000100 = B
00000100 = (A and B) > 0

00111011 = A
00000100 = B
00000000 = (A and B) = 0
```

To **set** bits in a bit pattern, we OR it with a pattern with a 1 only at the places of the bits that we want to set. Imagine we want to set the third least-significant bit:

```
00110001 = A
00000100 = B
00110101 = (A or B)→A

00110101 = A
00000100 = B
00110101 = (A or B)→A
```

To **reset** bits in a bit pattern, we AND it with a pattern with a 0 only at the places of the bits that we want to reset. Imagine we want to reset the third least-significant bit:

```
00110001 = A
11111011 = B
00110001 = (A and B)→A
```

```
00110101 = A
11111011 = B
00110001 = (A and B)→A
```

To **invert** bits in a bit pattern, we XOR it with a pattern with a 1 only at the places of the bits that we want to invert. Imagine we want to invert the third least-significant bit:

```
00110001 = A
00000100 = B
00110101 = (A xor B)→A

00110101 = A
00000100 = B
00110001 = (A xor B)→A
```

Exercise 1: Write a MIPS program that multiplies two numbers using the Russian peasant algorithm (shift-mask-add).
Exercise 2: Write a MIPS program that can do integer division using long-tail (shift-compare-subtract)

The relevant (new) instructions for today are:

`sll`	Shift left logic (immediate)
`srl`	Shift right logic (immediate)
`sra`	Shift right arithmetic (immediate)
`sllv`	Shift left logic (register)
`srlv`	Shift right logic (register)
`srav`	Shift right arithmetic (register)
`and`	Bitwise AND
`or`	Bitwise OR
`xor`	Bitwise XOR