

Intro. to Computer Architecture

Daniel Page

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB. UK.
(csdsp@bristol.ac.uk)

January 9, 2018

Keep in mind there are *two* PDFs available (of which this is the latter):

1. a PDF of examinable material used as lecture slides, and
2. a PDF of non-examinable, extra material:
 - ▶ the associated notes page may be pre-populated with extra, written explanation of material covered in lecture(s), plus
 - ▶ anything with a “grey’ed out” header/footer represents extra material which is useful and/or interesting but out of scope (and hence not covered).

Notes:

Notes:

Example

Assuming $ci = 0$, the approach you know sets $b = 10$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto \quad 1 \quad 0 \quad 7 \\ y & = & 14_{(10)} & \mapsto \quad \underline{0 \quad 1 \quad 4} + \\ c & = & & \\ r & = & & \end{array}$$

but it *also* applies for $b = 2$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ y & = & 14_{(10)} & \mapsto & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & + \\ c & = & & & \hline r & = & & & \hline \end{array}$$

Notes:

Addition in theory (1)

Example

Assuming $ci = 0$, the approach you know sets $b = 10$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 1 & 0 & 7 \\ y & = & 14_{(10)} & \mapsto & 0 & 1 & 4 + \\ c & = & & & \hline r & = & & & & & 0 \end{array}$$

but it *also* applies for $b = 2$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ y & = & 14_{(10)} & \mapsto & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & + \\ c & = & & & & & & & & & & & \\ r & = & & & & & & & & & & & \end{array}$$

Notes:

© Daniel Page (csdsp@bristol.ac.uk)

Notes:

Notes:

Addition in theory (1)

Example

Assuming $ci = 0$, the approach you know sets $b = 10$

x

$=$

$107_{(10)}$

\mapsto

$1\ 0\ 7$

y

$=$

$14_{(10)}$

\mapsto

$0\ 1\ 4\ +$

c

$=$

\mapsto

$0\ 0\ 1\ 0$

r

$=$

$121_{(10)}$

\mapsto

$1\ 2\ 1$

but it *also* applies for $b = 2$

x

$=$

$107_{(10)}$

\mapsto

$0\ 1\ 1\ 0\ 1\ 0\ 1\ 1$

y

$=$

$14_{(10)}$

\mapsto

$0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ +$

c

$=$

\mapsto

r

$=$

\mapsto

Notes:

Addition in theory (1)

Example

Assuming $ci = 0$, the approach you know sets $b = 10$

x

$=$

$107_{(10)}$

\mapsto

$1\ 0\ 7$

y

$=$

$14_{(10)}$

\mapsto

$0\ 1\ 4\ +$

c

$=$

\mapsto

$0\ 0\ 1\ 0$

r

$=$

$121_{(10)}$

\mapsto

$1\ 2\ 1$

but it *also* applies for $b = 2$

x

$=$

$107_{(10)}$

\mapsto

$0\ 1\ 1\ 0\ 1\ 0\ 1\ 1$

y

$=$

$14_{(10)}$

\mapsto

$0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ +$

c

$=$

\mapsto

0

r

$=$

\mapsto

Notes:

Addition in theory (1)

Example

Assuming $ci = 0$, the approach you know sets $b = 10$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 1 & 0 & 7 \\ y & = & 14_{(10)} & \mapsto & 0 & 1 & 4 & + \\ c & = & & & 0 & 0 & 1 & 0 \\ r & = & 121_{(10)} & \mapsto & 1 & 2 & 1 \end{array}$$

but it *also* applies for $b = 2$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ y & = & 14_{(10)} & \mapsto & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & + \\ c & = & & & & & & & & 0 & 0 \\ r & = & & & & & & & & & 1 \end{array}$$

Notes:

Addition in theory (1)

Example

Assuming $ci = 0$, the approach you know sets $b = 10$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 1 & 0 & 7 \\ y & = & 14_{(10)} & \mapsto & 0 & 1 & 4 & + \\ c & = & & & 0 & 0 & 1 & 0 \\ r & = & 121_{(10)} & \mapsto & 1 & 2 & 1 \end{array}$$

but it *also* applies for $b = 2$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ y & = & 14_{(10)} & \mapsto & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & + \\ c & = & & & & & & & 1 & 0 & 0 \\ r & = & & & & & & & & 0 & 1 \end{array}$$

Notes:

Addition in theory (1)

Example

Assuming $ci = 0$, the approach you know sets $b = 10$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 1 & 0 & 7 \\ y & = & 14_{(10)} & \mapsto & 0 & 1 & 4 & + \\ c & = & & & 0 & 0 & 1 & 0 \\ r & = & 121_{(10)} & \mapsto & 1 & 2 & 1 \end{array}$$

but it *also* applies for $b = 2$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ y & = & 14_{(10)} & \mapsto & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & + \\ c & = & & & & & & & 1 & 1 & 0 & 0 \\ r & = & & & & & & & & 0 & 0 & 1 \end{array}$$

Notes:

Addition in theory (1)

Example

Assuming $ci = 0$, the approach you know sets $b = 10$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 1 & 0 & 7 \\ y & = & 14_{(10)} & \mapsto & 0 & 1 & 4 & + \\ c & = & & & 0 & 0 & 1 & 0 \\ r & = & 121_{(10)} & \mapsto & 1 & 2 & 1 \end{array}$$

but it *also* applies for $b = 2$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ y & = & 14_{(10)} & \mapsto & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & + \\ c & = & & & & & & & 1 & 1 & 1 & 0 \\ r & = & & & & & & & & 1 & 0 & 0 & 1 \end{array}$$

Notes:

Addition in theory (1)

Example

Assuming $ci = 0$, the approach you know sets $b = 10$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 1 & 0 & 7 \\ y & = & 14_{(10)} & \mapsto & 0 & 1 & 4 & + \\ c & = & & & 0 & 0 & 1 & 0 \\ r & = & 121_{(10)} & \mapsto & 1 & 2 & 1 \end{array}$$

but it *also* applies for $b = 2$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ y & = & 14_{(10)} & \mapsto & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & + \\ c & = & & & 0 & 1 & 1 & 1 & 0 & 0 & & \\ r & = & & & 1 & 1 & 0 & 0 & 1 & & & \end{array}$$

Notes:

Addition in theory (1)

Example

Assuming $ci = 0$, the approach you know sets $b = 10$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 1 & 0 & 7 \\ y & = & 14_{(10)} & \mapsto & 0 & 1 & 4 & + \\ c & = & & & 0 & 0 & 1 & 0 \\ r & = & 121_{(10)} & \mapsto & 1 & 2 & 1 \end{array}$$

but it *also* applies for $b = 2$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ y & = & 14_{(10)} & \mapsto & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & + \\ c & = & & & 0 & 0 & 1 & 1 & 1 & 0 & 0 & \\ r & = & & & 1 & 1 & 1 & 0 & 0 & 1 & & \end{array}$$

Notes:

Addition in theory (1)

Example

Assuming $ci = 0$, the approach you know sets $b = 10$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 1 & 0 & 7 \\ y & = & 14_{(10)} & \mapsto & 0 & 1 & 4 & + \\ c & = & & & 0 & 0 & 1 & 0 \\ r & = & 121_{(10)} & \mapsto & 1 & 2 & 1 \end{array}$$

but it *also* applies for $b = 2$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ y & = & 14_{(10)} & \mapsto & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & + \\ c & = & & & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ r & = & & & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}$$

Notes:

Addition in theory (1)

Example

Assuming $ci = 0$, the approach you know sets $b = 10$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 1 & 0 & 7 \\ y & = & 14_{(10)} & \mapsto & 0 & 1 & 4 & + \\ c & = & & & 0 & 0 & 1 & 0 \\ r & = & 121_{(10)} & \mapsto & 1 & 2 & 1 \end{array}$$

but it *also* applies for $b = 2$

$$\begin{array}{rcll} x & = & 107_{(10)} & \mapsto & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ y & = & 14_{(10)} & \mapsto & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & + \\ c & = & & & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ r & = & 121_{(10)} & \mapsto & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}$$

Notes:

Algorithm

Input: Two unsigned, n -digit, base- b integers x and y , and a 1-digit carry-in $ci \in \{0, 1\}$

Output: An unsigned, n -digit, base- b integer $r = x + y$, and a 1-digit carry-out $co \in \{0, 1\}$


```
1  $r \leftarrow 0, c_0 \leftarrow ci$ 
2 for  $i = 0$  upto  $n - 1$  step  $+1$  do
3    $r_i \leftarrow (x_i + y_i + c_i) \bmod b$ 
4   if  $(x_i + y_i + c_i) < b$  then  $c_{i+1} \leftarrow 0$  else  $c_{i+1} \leftarrow 1$ 
5 end
6  $co \leftarrow c_n$ 
7 return  $r, co$ 
```

Notes:

- Certain (subtle) features of the algorithm should be kept in mind:
 - adding two n -digit integers produces an $(n + 1)$ -digit result,
 - instead of producing this directly (per the example), here the outputs are produced as a *separate* sum and carry-out,
 - although the carry-in *could* be implicitly zero, here it is an explicit extra input,
 - although the algorithm *looks* like it's got a loop in it, we can unroll this since n is fixed, and
 - when $b = 2$, the body of the loop is a Boolean function: each of the inputs and outputs is a value in the set $\{0, 1\}$.

Definition

The behaviour of the component

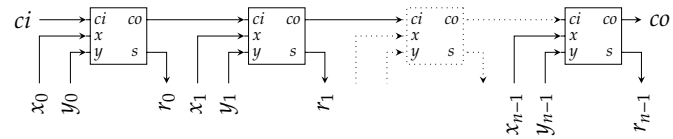


is described by the truth table

FULL-ADDER				
ci	x	y	co	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Notes:

Circuit



Notes:

An Aside: Carry-out versus overflow

- ▶ The magnitude of a result is too large to represent in n bits if *either*
 1. the inputs are *unsigned* and there is a carry-out, *or*
 2. the inputs are *signed* but the sign of the result makes no sense
 which are termed **carry-out** and **overflow** conditions.
- ▶ To cope, two steps are typically applied:
 1. detect the condition, then
 2. take some action, e.g.,
 - ▶ signal the condition somehow (e.g., via a status register or some form of exception) but **truncate** the result to n bits, or
 - ▶ **clamp** (or **saturate**) the result to the largest magnitude representable in n bits to “fix” it.

Notes:

Example

Consider use of an *unsigned* representation:

$$\begin{array}{rcll} x & = & 15_{(10)} & \mapsto & \begin{array}{cccc} 1 & 1 & 1 & 1 \end{array} \\ y & = & 1_{(10)} & \mapsto & \begin{array}{cccc} 0 & 0 & 0 & 1 \end{array} \\ c & = & & & \begin{array}{cccc} 1 & 1 & 1 & 0 \end{array} \\ r & = & 0_{(10)} & \mapsto & \begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \end{array} +$$

Here, the carry-out indicates an error: the correct result $r = 16$ is too large for $n = 4$ bits.

Notes:

Example

Consider use of a *signed* representation:

$$\begin{array}{rcll} x & = & -1_{(10)} & \mapsto & \begin{array}{cccc} 1 & 1 & 1 & 1 \end{array} \\ y & = & 1_{(10)} & \mapsto & \begin{array}{cccc} 0 & 0 & 0 & 1 \end{array} \\ c & = & & & \begin{array}{cccc} 1 & 1 & 1 & 0 \end{array} \\ r & = & 0_{(10)} & \mapsto & \begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \end{array} +$$

Irrespective of the carry-out, the signs of inputs and output make sense: there is no overflow, so $r = 0$ is correct.

Example

Consider use of a *signed* representation:

$$\begin{array}{rcll} x & = & 7_{(10)} & \mapsto & \begin{array}{cccc} 0 & 1 & 1 & 1 \end{array} \\ y & = & 1_{(10)} & \mapsto & \begin{array}{cccc} 0 & 0 & 0 & 1 \end{array} \\ c & = & & & \begin{array}{cccc} 0 & 1 & 1 & 0 \end{array} \\ r & = & -8_{(10)} & \mapsto & \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array} \end{array} +$$

Irrespective of the carry-out, the signs of inputs and output make no sense: there is an overflow, so $r = -8$ is incorrect.

- Detection of an overflow condition is slightly more complicated than for carry; in computation of $r = x + y$, we need to check

x +ve	y -ve	\Rightarrow	no overflow
x -ve	y +ve	\Rightarrow	no overflow
x +ve	y +ve	r +ve	\Rightarrow no overflow
x +ve	y +ve	r -ve	\Rightarrow overflow
x -ve	y -ve	r +ve	\Rightarrow overflow
x -ve	y -ve	r -ve	\Rightarrow no overflow

for example.

Notes:

Conclusions

► Take away points:

1. The design strategy we used is important and (fairly) general-purpose:
 - 1.1 explore and understand an approach in theory,
 - 1.2 translate, formalise, and generalise the approach into an algorithm,
 - 1.3 translate the algorithm into a hardware design,
 - 1.4 refine (or select) the hardware design to satisfy any design constraints.
2. Computer arithmetic is an interesting special-case:
 - it's a broad topic with a rich history,
 - there's usually a large design space of potential approaches,
 - they're often easy to understand at an intuitive, high level,
 - correctness and efficiency of resulting low-level solutions is vital and challenging.

Notes:

Additional Reading

- *Wikipedia: Computer Arithmetic*. URL: http://en.wikipedia.org/wiki/Category:Computer_arithmetic.
- D. Page. “Chapter 7: Arithmetic and logic”. In: *A Practical Introduction to Computer Architecture*. 1st ed. Springer-Verlag, 2009.
- B. Parhami. “Part 2: Addition/subtraction”. In: *Computer Arithmetic: Algorithms and Hardware Designs*. 1st ed. Oxford University Press, 2000.
- W. Stallings. “Chapter 10: Computer arithmetic”. In: *Computer Organisation and Architecture*. 9th ed. Prentice-Hall, 2013.
- A.S. Tanenbaum and T. Austin. “Section 3.2.2: Arithmetic circuits”. In: *Structured Computer Organisation*. 6th ed. Prentice-Hall, 2012.

Notes:

References

- [1] [Wikipedia: Computer Arithmetic](http://en.wikipedia.org/wiki/Category:Computer_arithmetic). URL: http://en.wikipedia.org/wiki/Category:Computer_arithmetic (see p. 51).
- [2] D. Page. “Chapter 7: Arithmetic and logic”. In: *A Practical Introduction to Computer Architecture*. 1st ed. Springer-Verlag, 2009 (see p. 51).
- [3] B. Parhami. “Part 2: Addition/subtraction”. In: *Computer Arithmetic: Algorithms and Hardware Designs*. 1st ed. Oxford University Press, 2000 (see p. 51).
- [4] W. Stallings. “Chapter 10: Computer arithmetic”. In: *Computer Organisation and Architecture*. 9th ed. Prentice-Hall, 2013 (see p. 51).
- [5] A.S. Tanenbaum and T. Austin. “Section 3.2.2: Arithmetic circuits”. In: *Structured Computer Organisation*. 6th ed. Prentice-Hall, 2012 (see p. 51).

Notes: