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 explaination 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:

- Goal: implement a design(s) for integer multiplication, i.e.,
  - accepts n-bit
    - y, the **multiplier** that does the multiplying, and
    - x, the **multiplicand** that is multiplied

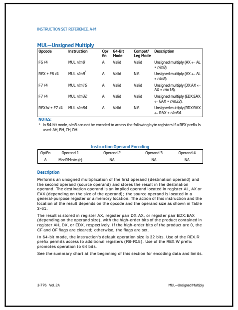
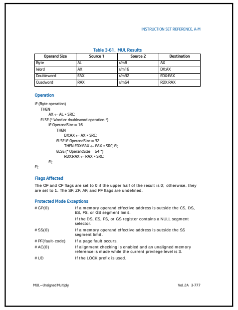
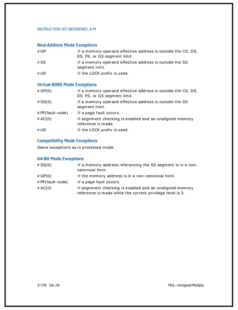
    as input, and
  - produces $2n$-bit **product** $r = y \cdot x$ as output

  which

  1. functions correctly, *and*
  2. is efficient (wrt. whatever metrics are deemed important).

Notes:

COMS12200 lecture: week #7

Notes:

Multiplication in theory (1)

Approach #1: long-hand multiplication

Example

Consider a case where $b = 10$, $x = 623_{(10)}$ and $y = 567_{(10)}$:

x	$=$	$623_{(10)}$	$\mapsto$				6	2	3
y	$=$	$567_{(10)}$	$\mapsto$				5	6	7 $\times$
p_0	$= 7 \cdot 3 \cdot 10^0 =$	$21_{(10)}$	$\mapsto$					2	1
p_1	$= 7 \cdot 2 \cdot 10^1 =$	$140_{(10)}$	$\mapsto$				1	4	
p_2	$= 7 \cdot 6 \cdot 10^2 =$	$4200_{(10)}$	$\mapsto$			4	2		
p_3	$= 6 \cdot 3 \cdot 10^1 =$	$180_{(10)}$	$\mapsto$				1	8	
p_4	$= 6 \cdot 2 \cdot 10^2 =$	$1200_{(10)}$	$\mapsto$			1	2		
p_5	$= 6 \cdot 6 \cdot 10^3 =$	$36000_{(10)}$	$\mapsto$		3	6			
p_6	$= 5 \cdot 3 \cdot 10^2 =$	$1500_{(10)}$	$\mapsto$			1	5		
p_7	$= 5 \cdot 2 \cdot 10^3 =$	$10000_{(10)}$	$\mapsto$		1	0			
p_8	$= 5 \cdot 6 \cdot 10^4 =$	$300000_{(10)}$	$\mapsto$	3	0				
r	$=$	$353241_{(10)}$	$\mapsto$	3	5	3	2	4	1

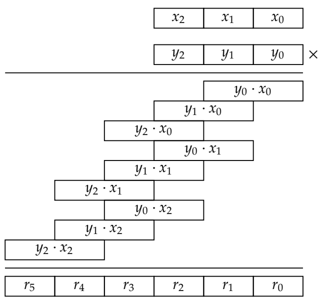
Notes:

Multiplication in theory (2)

Approach #1: long-hand multiplication

Example (operand scanning)

Consider an example where where $|x| = |y| = 3$:

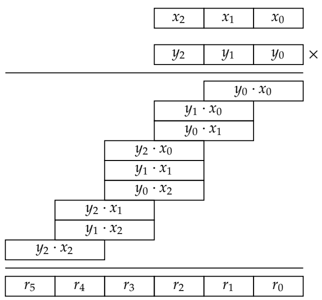


Notice that

1. an outer-loop steps through digits of y, say y_i,
2. an inner-loop steps through digits of x, say x_j.

Example (product scanning)

Consider an example where where $|x| = |y| = 3$:



Notice that

1. an outer-loop steps through digits of r, say r_i,
2. two inner-loops step through matching digits of x and y, say x_j and x_i.

Notes:

Algorithm (operand scanning)

Input: Two unsigned, base-b integers x and y
Output: An unsigned, base-b integer $r = x \cdot y$

```
l_x ← |x|, l_y ← |y|, l_r ← l_x + l_y
r ← 0
for j = 0 upto l_y − 1 step +1 do
    c ← 0
    for i = 0 upto l_x − 1 step +1 do
        u · b + v = t ← y_j · x_i + r_{j+i} + c
        r_{j+i} ← v
        c ← u
    end
    r_{j+l_x} ← c
end
return r
```

Notes:

Example (operand scanning)

Consider a case where $b = 10$, $x = 623_{(10)}$ and $y = 567_{(10)}$:

j	i	r	c	y_i	x_j	$t = y_i \cdot x_i + r_{i+j} + c$	r'	c'
		$\langle 0,0,0,0,0,0 \rangle$						
0	0	$\langle 0,0,0,0,0,0 \rangle$	0	7	3	21	$\langle 1,0,0,0,0,0 \rangle$	2
0	1	$\langle 1,0,0,0,0,0 \rangle$	2	7	2	16	$\langle 1,6,0,0,0,0 \rangle$	1
0	2	$\langle 1,6,0,0,0,0 \rangle$	1	7	6	43	$\langle 1,6,3,0,0,0 \rangle$	4
0		$\langle 1,6,3,0,0,0 \rangle$	4				$\langle 1,6,3,4,0,0 \rangle$	
1	0	$\langle 1,6,3,4,0,0 \rangle$	0	6	3	24	$\langle 1,4,3,4,0,0 \rangle$	2
1	1	$\langle 1,4,3,4,0,0 \rangle$	2	6	2	17	$\langle 1,4,7,4,0,0 \rangle$	1
1	2	$\langle 1,4,7,4,0,0 \rangle$	1	6	6	41	$\langle 1,4,7,1,0,0 \rangle$	4
1		$\langle 1,4,7,1,0,0 \rangle$	4				$\langle 1,4,7,1,4,0 \rangle$	
2	0	$\langle 1,4,7,1,4,0 \rangle$	0	5	3	22	$\langle 1,4,2,1,4,0 \rangle$	2
2	1	$\langle 1,4,2,1,4,0 \rangle$	2	5	2	13	$\langle 1,4,2,3,4,0 \rangle$	1
2	2	$\langle 1,4,2,3,5,0 \rangle$	1	5	6	35	$\langle 1,4,2,3,5,0 \rangle$	3
2		$\langle 1,4,2,3,5,0 \rangle$	3				$\langle 1,4,2,3,5,3 \rangle$	3
		$\langle 1,4,2,3,5,3 \rangle$						

Notes:

Algorithm (product scanning)

Input: Two unsigned, base-b integers x and y
Output: An unsigned, base-b integer $r = x \cdot y$

```
l_x ← |x|, l_y ← |y|, l_r ← l_x + l_y
r ← 0, c_0 ← 0, c_1 ← 0, c_2 ← 0
for k = 0 upto l_x + l_y − 1 step +1 do
    for j = 0 upto l_y − 1 step +1 do
        for i = 0 upto l_x − 1 step +1 do
            if (j + i) = k then
                u · b + v = t ← y_j · x_i
                c · b + c_0 = t ← c_0 + v
                c · b + c_1 = t ← c_1 + u + c
                c_2 ← c_2 + c
            end
        end
    end
    r_k ← c_0, c_0 ← c_1, c_1 ← c_2, c_2 ← 0
end
r_{l_x+l_y−1} ← c_0
```

Notes:

Example (product scanning)

Consider a case where $b = 10$, $x = 623_{(10)}$ and $y = 567_{(10)}$:

k	j	i	r	c_2	c_1	c_0	y_i	x_j	$t = y_i \cdot x_i$	r'	c'_2	c'_1	c'_0
			$\langle 0,0,0,0,0,0 \rangle$	0	0	0							
0	0	0	$\langle 0,0,0,0,0,0 \rangle$	0	0	0	7	3	21	$\langle 0,0,0,0,0,0 \rangle$	0	2	1
0			$\langle 0,0,0,0,0,0 \rangle$	0	2	1				$\langle 1,0,0,0,0,0 \rangle$	0	0	2
1	0	1	$\langle 1,0,0,0,0,0 \rangle$	0	0	2	7	2	14	$\langle 1,0,0,0,0,0 \rangle$	0	1	6
1	1	0	$\langle 1,0,0,0,0,0 \rangle$	0	1	6	6	3	18	$\langle 1,0,0,0,0,0 \rangle$	0	3	4
1			$\langle 1,0,0,0,0,0 \rangle$	0	3	4				$\langle 1,4,0,0,0,0 \rangle$	0	0	3
2	0	2	$\langle 1,4,0,0,0,0 \rangle$	0	0	3	7	6	42	$\langle 1,4,0,0,0,0 \rangle$	0	4	5
2	1	1	$\langle 1,4,0,0,0,0 \rangle$	0	4	5	6	2	12	$\langle 1,4,0,0,0,0 \rangle$	0	5	7
2	2	0	$\langle 1,4,0,0,0,0 \rangle$	0	4	7	5	3	15	$\langle 1,4,0,0,0,0 \rangle$	0	7	2
2			$\langle 1,4,0,0,0,0 \rangle$	0	7	2				$\langle 1,4,2,0,0,0 \rangle$	0	0	7
3	1	2	$\langle 1,4,2,0,0,0 \rangle$	0	0	7	6	6	36	$\langle 1,4,2,0,0,0 \rangle$	0	4	3
3	2	1	$\langle 1,4,2,0,0,0 \rangle$	0	4	3	5	2	10	$\langle 1,4,2,0,0,0 \rangle$	0	5	3
3			$\langle 1,4,2,0,0,0 \rangle$	0	5	3				$\langle 1,4,2,3,0,0 \rangle$	0	0	5
4	2	2	$\langle 1,4,2,3,0,0 \rangle$	0	0	5	5	6	30	$\langle 1,4,2,3,0,0 \rangle$	0	3	5
4			$\langle 1,4,2,3,0,0 \rangle$	0	3	5				$\langle 1,4,2,3,5,0 \rangle$	0	0	3
			$\langle 1,4,2,3,5,0 \rangle$	0	0	3				$\langle 1,4,2,3,5,3 \rangle$	0	0	3
			$\langle 1,4,2,3,5,3 \rangle$										

Notes:

Multiplication in theory (7)
Approach #2: multiplication as repeated addition

- Idea: scalar multiplication of x by y is simply repeated addition, i.e.,

$$y \cdot x = \underbrace{x + x + \cdots + x + x}_{y \text{ terms}},$$

so, if $y = 14_{(10)}$, then

$$14 \cdot x = x + x + x + x + x + x + x + x + x + x + x + x + x + x.$$

Notes:

Multiplication in theory (8)
Approach #2: multiplication as repeated addition

- Idea: by writing y in base-b we express the product as a summation, i.e.,

$$y \cdot x \equiv \left(\sum_{i=0}^{n-1} y_i \cdot b^i\right) \cdot x \equiv \sum_{i=0}^{n-1} y_i \cdot x \cdot b^i$$

so, if $y = 14_{(10)} \mapsto 1110_{(2)}$, then

$$\begin{array}{ccccccccc} y \cdot x & = & y_0 \cdot x \cdot 2^0 & + & y_1 \cdot x \cdot 2^1 & + & y_2 \cdot x \cdot 2^2 & + & y_3 \cdot x \cdot 2^3 \\ & = & 0 \cdot x \cdot 2^0 & + & 1 \cdot x \cdot 2^1 & + & 1 \cdot x \cdot 2^2 & + & 1 \cdot x \cdot 2^3 \\ & = & 0 \cdot x & + & 2 \cdot x & + & 4 \cdot x & + & 8 \cdot x \\ & = & 14 \cdot x & & & & & & \end{array}$$

noting that *if* $b = 2$, for any t

1. we know

$$t \cdot 2^i \equiv t \ll i,$$

and

2. we know

$$t \cdot y_i \equiv \begin{cases} t & \text{if } y_i = 1 \\ 0 & \text{if } y_i = 0 \end{cases}$$

Notes:

Multiplication in practice (1)
Candidate design #2: combinatorial, bit-parallel

- Idea: for $b = 2$ and given

$$\sum_{i=0}^{n-1} y_i \cdot x \cdot 2^i,$$

construct a combinatorial design to compute r via

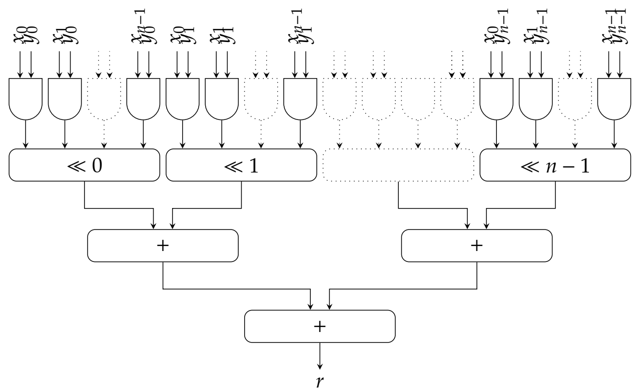
1. some AND gates to generate partial products (i.e., $y_i \cdot x$),
2. some left-shift components to scale the partial products correctly (i.e., $y_i \cdot x \cdot 2^i$), and
3. some adder components to sum the scaled partial products

i.e., ...

Notes:

Multiplication in practice (2)
Candidate design #2: combinatorial, bit-parallel

Circuit



- Note that this design

−ve: requires a larger data-path
+ve: requires a smaller control-path (i.e., none at all),
+ve: requires less steps (i.e., 1),
−ve: has a longer critical path (meaning each step is longer).

Notes:

Multiplication in practice (3)

Candidate design #2: iterative, bit-serial

- Idea: for $b = 2$, apply **Horner's rule** st. if $y = 14_{(10)} \mapsto 1110_{(2)}$ we (re)bracket

$$\sum_{i=0}^{n-1} y_i \cdot x \cdot 2^i$$

as

$$\begin{array}{rcl}
y \cdot x & = & y_0 \cdot x + 2 \cdot (\; y_1 \cdot x + 2 \cdot (\; y_2 \cdot x + 2 \cdot (\; y_3 \cdot x + 2 \cdot (\, 0 \,) \,) \,) \,) \\
& = & 0 \cdot x + 2 \cdot (\; 1 \cdot x + 2 \cdot (\; 1 \cdot x + 2 \cdot (\; 1 \cdot x + 2 \cdot (\, 0 \,) \,) \,) \,) \\
& = & 0 \cdot x + 2 \cdot (\; 1 \cdot x + 2 \cdot (\; 1 \cdot x + 2 \cdot (\; 1 \cdot x + \quad 0 \quad) \,) \,) \\
& = & 0 \cdot x + 2 \cdot (\; 1 \cdot x + 2 \cdot (\; 1 \cdot x + 2 \cdot (\; 1 \cdot x \quad) \,) \,) \\
& = & 0 \cdot x + 2 \cdot (\; 1 \cdot x + 2 \cdot (\; 1 \cdot x + \quad 2 \cdot x \quad) \,) \\
& = & 0 \cdot x + 2 \cdot (\; 1 \cdot x + 2 \cdot (\; 3 \cdot x \quad) \,) \\
& = & 0 \cdot x + 2 \cdot (\; 1 \cdot x + \quad 6 \cdot x \quad) \\
& = & 0 \cdot x + 2 \cdot (\; 7 \cdot x \quad) \\
& = & 0 \cdot x + \quad 14 \cdot x \\
& = & 14 \cdot x
\end{array}$$

then ...

Notes:

Multiplication in practice (4)

Candidate design #2: iterative, bit-serial

- ... evaluate the expression inside-out, *accumulating* the result via the rule

$$r \leftarrow y_i \cdot x + 2 \cdot r$$

i.e.,

Algorithm

Input: Two unsigned, n-bit, base-2 integers x and y
Output: An unsigned, $2n$-bit, base-2 integer $r = y \cdot x$

```
r ← 0
for i = n − 1 downto 0 step −1 do
    r ← 2 · r
    if y_i = 1 then
        r ← r + x
    end
end
return r
```

Example

Consider a case where $y = 14_{(10)} \mapsto 1110_{(2)}$:

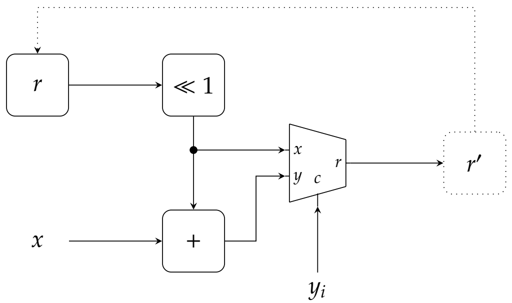
i	r	y_i	r'	
	0			
3	0	1	x	$r' \leftarrow 2 \cdot r + x$
2	x	1	$3 \cdot x$	$r' \leftarrow 2 \cdot r + x$
1	$3 \cdot x$	1	$7 \cdot x$	$r' \leftarrow 2 \cdot r + x$
0	$7 \cdot x$	0	$14 \cdot x$	$r' \leftarrow 2 \cdot r$
	$14 \cdot x$			

Notes:

Multiplication in practice (5)

Candidate design #2: iterative, bit-serial

Circuit



- Note that this design

  +ve: requires a smaller data-path
  −ve: requires a larger control-path (i.e., an entire FSM),
  −ve: requires more steps (i.e., n),
  +ve: has a shorter critical path (meaning each step is shorter).

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 3: Multiplication". 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*. URL: `http://en.wikipedia.org/wiki/Category:Computer_arithmetic` (see p. 37).

[2] D. Page. "Chapter 7: Arithmetic and logic". In: *A Practical Introduction to Computer Architecture*. 1st ed. Springer-Verlag, 2009 (see p. 37).

[3] B. Parhami. "Part 3: Multiplication". In: *Computer Arithmetic: Algorithms and Hardware Designs*. 1st ed. Oxford University Press, 2000 (see p. 37).

[4] W. Stallings. "Chapter 10: Computer arithmetic". In: *Computer Organisation and Architecture*. 9th ed. Prentice-Hall, 2013 (see p. 37).

[5] A.S. Tanenbaum and T. Austin. "Section 3.2.2: Arithmetic circuits". In: *Structured Computer Organisation*. 6th ed. Prentice-Hall, 2012 (see p. 37).

Notes: