CTA01 – Huiswerk week 4

Vandaag hebben we paragraaf 3.1, 3.2, 3.3 en 3.5 van het boek behandeld. De slides staan op de de wiki in pptx en pdf formaat. We hebben van H3 slide 1 t/m 31 behandeld.

Huiswerk

Lees paragraaf 3.1, 3.2, 3.3 en 3.5 door. Vragen over de tekst graag uiterlijk volgende lesweek dinsdag 23:59 uur mailen zodat ik ze op woensdag in de les kan behandelen. Vergeet niet om de 'check yourself' oefeningen te maken. De antwoorden staan op p. 253.

Maak de programmeeropdrachten op slide 27 en slide 28! De uitwerkingen staan ook op de slides.

Breng de volgende verandering aan bij de opgave in je boek:

3.15 Er wordt verwezen naar de tekst op p. 196. Vervang "31" door "63" in de opgave. In de tekst op p. 196 staat "64" maar dat moet dus ook "63" zijn.

Maak de volgende opgaven uit paragraaf 3.13: 3.13 t/m 3.16, 3.22 t/m 3.24 en 3.41.

Vragen over de opgaven graag uiterlijk volgende lesweek dinsdag 23:59 uur mailen zodat ik ze op woensdag in de les kan behandelen. Maak een issue aan op de wiki als je een fout in dit document hebt gevonden.

Mijn antwoorden

3.13 `0x62` × `0x12` (= `0x06E4`)
multiplicant = `01100010`
See Table 1.

3.14 For hardware, it takes one cycle to do the add, one cycle to do the shift, and one cycle to decide if we are done. So the loop takes (3 × 8) cycles,

Table 1: Stepwise execution of the multiplication.

step	action	product/multiplier
0	init	0000000000010010
1a	lsb = 0 ⇒ NOP	0000000000010010
1b	LSR	0000000000001001
2a	lsb = 1 ⇒ product = product + multiplicant	0110001000001001
2b	LSR	0011000100000100
3a	lsb = 0 ⇒ NOP	0011000100000100
3b	LSR	0001100010000010
4a	lsb = 0 ⇒ NOP	0001100010000010
4b	LSR	0000110001000001
5a	lsb = 1 ⇒ product = product + multiplicant	0110111001000001
5b	LSR	0011011100100000
6a	lsb = 0 ⇒ NOP	0011011100100000
6b	LSR	0001101110010000
7a	lsb = 0 ⇒ NOP	0001101110010000
7b	LSR	0000110111001000
8a	lsb = 0 ⇒ NOP	0000110111001000
8b	LSR	0000011011100100

with each cycle being 4 time units long. $(3 \times 8) \times 4\,\text{tu} = 96$ time units for hardware.

For a software implementation, it takes one cycle to decide what to add, one cycle to do the add, one cycle to do each shift , and one cycle to decide if we are done. So the loop takes (5×8) cycles, with each cycle being 4 time units long. $(5 \times 8) \times 4\,\text{tu} = 160$ time units for software.

3.15 It takes 4 time units to get through an adder, and there will be $8-1=7$ adders. $7 \times 4\,\text{tu} = 28$ time units.

3.16 It takes 4 time units to get through an adder, and the adders are arranged in a tree structure. It will require ${}^{2}\log 8 = 3$ levels. $3 \times 4\,\text{tu} = 12$ time units.

3.22 `0x0C000000` = `0000 1100 0000 0000 0000 0000 0000 0000`
sign is positive
exp = `0x18` $= 24-127 = -103$
there is a hidden 1
mantissa = 0
answer $= 1.0 \times 2^{-103} = 9.860\,761 \times 10^{-32}$

3.23 $63.25 = 111111.01$
normalize, move binary point five to the left
1.1111101×2^{5}
sign = positive, exp $= 127 + 5 = 132$
Final bit pattern:
`0100 0010 0111 1101 0000 0000 0000 0000` = `0x427D0000`

3.24 $63.25 = 111111.01$
normalize, move binary point five to the left
$1.1111101x2^{5}$
sign = positive, exp $= 1023 + 5 = 1028$
Final bit pattern:
`0100 0000 0100 1111 1010 0000 0000 0000 0000 0000 0000 0000`
`0000 0000 0000 0000 = 0x404FA00000000000`

3.41 `1011 1110 1000 0000 0000 0000 0000 0000 0000`
`0xBE800000`
Exact