CTA01 – Huiswerk week 3

Vandaag hebben we paragraaf 2.8, 2.10, 2.13 en 2.14 van het boek behandeld. De slides staan op de de wiki in pptx en pdf formaat. We hebben van H2 slide 50 t/m 83 behandeld.

Huiswerk

Lees paragraaf 2.8, 2.10, 2.13 en 2.14 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. 184.

Breng de volgende veranderingen aan bij de opgaven in je boek:

2.30 The hint is **wrong**. It should be removed. Assume `long long int` parameter and return value.

2.32 Again assume `long long int` instead of `int`.

Maak de volgende opgaven uit paragraaf 2.23: 2.19, 2.23, 2.24, 2.30, 2.32, 2.34, 2.37, 2.41 en 2.42.

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

2.19

```
ORRI    X12, XZR, #0x3F
LSL     X12, X12, #11
AND     X13, X10, X12
LSL     X13, X13, #15
MOVZ    X12, #0xFC00, LSL #16
SUBI    X14, XZR, #1
EOR     X12, X12, X14
AND     X11, X11, X12
```

```
    ORR     X11, X11, X13
```

2.23.1 [0x0000000018000000, 0x0000000027FFFFFC]

2.23.2 [0x000000001FF00000, 0x00000000200FFFFC]

2.24.1 The CB instruction format would be most appropriate because it would allow the maximum number of bits possible for the "loop" parameter, thereby maximizing the utility of the instruction.

2.24.2 Voor de hand ligt:

```
    SUBIS   XZR, X12, #0
    B.LE    not_loop
    SUBI    X12, X12, #1
    B       loop
not_loop:
```

Maar korter is:

```
    SUBIS   X12, X12, #1
    B.GE    loop
    ADDI    X12, X12, #1
```

2.30 The hint is **wrong**. It should be removed. Assume `long long int` parameter and return value. Otherwise `LDURW` is needed which we skipped.

```
fib:
    CBZ     X0, done
    CMPI    X0, #1
    B.EQ    done
    SUBI    SP, SP, #24
    STUR    X0, [SP, #0]
    STUR    LR, [SP, #8]
    STUR    X19, [SP, #16]
    SUBI    X0, X0, #1
    BL      fib
    ADDI    X19, X0, #0
    LDUR    X0, [SP, #0]
    SUBI    X0, X0, #2
    BL      fib
```

```
        ADD     X0, X0, X19
        LDUR    LR, [SP, #8]
        LDUR    X19, [SP, #16]
        ADDI    SP, SP, #24
done:
        BR      LR
```

2.32 Again assume **long long int** instead of **int**

```
f:
        SUBI    SP, SP, #16
        STUR    X19, [SP, #0]
        STUR    LR, [SP, #8]
        ADD     X19, X2, X3
        BL      g
        MOV     X1, X19
        BL      g
        LDUR    X19, [SP, #0]
        LDUR    LR, [SP, #8]
        ADDI    SP, SP, #16
        BR      LR
```

2.34 We have no idea what the contents of X5 are, g can set X5 as it pleases. We don't know what the precise contents of X29 and SP are; but we do know that they are identical to the contents when f was called. Similarly, we don't know what the precise contents of X30 are; but, we do know that it is equal to the return address set by the **BL** f instruction that invoked f.

2.37

```
        MOVZ    X0, #0x7788
        MOVK    X0, #0x5566, LSL #16
        MOVK    X0, #0x3344, LSL #32
        MOVK    X0, #0x1122, LSL #48
```

Or for the lazy student:

```
        LDA     0x1122334455667788
```

2.41.1 No.

2.41.2 If we double the performance of arithmetic instructions, then the speedup will be 1.07. If we improve the performance of arithmetic instructions by a factor of 10, then the speedup will be 1.13.

2.42.1 Average CPI = 2.6

2.42.2 For a 25 % improvement, the arithmetic instructions must have a CPI of at most 1.07.

2.42.3 For a 50 % improvement, the arithmetic instructions must have a CPI of at most 0.14.