CTA01 – Huiswerk week 2

Vandaag hebben we paragraaf 2.1 t/m 2.7 van het boek behandeld. De slides staan op de de wiki in pptx en pdf formaat. We hebben van H2 slide 1 t/m 49 behandeld.

Huiswerk

Lees paragraaf 2.1 t/m 2.7 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.

Maak de volgende opgaven uit paragraaf 2.23: 2.1 t/m 2.4, 2.8 t/m 2.11, 2.13, 2.14, 2.18, 2.20 t/m 2.22, 2.25 t/m 2.29.

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

2.1

```
SUBI    X9, X2, #5
ADD     X0, X1, X9
```

2.2

```
f = g + h + i;
```

2.3

```
SUB     X9, X3, X4 // compute i-j
LSL     X9, X9, #3 // multiply by 8 to convert ↩
↪ the doubleword offset to a byte offset
ADD     X11, X6, X9 // compute &A[i-j]
LDUR    X10, [X11, #0] // load A[i-j]
STUR    X10, [X7, #64] // store in B[8]
```

2.4

```
B[g] = A[f] + A[f+1]; f = A[f]; // notice the ↩
↪ order!
```

2.8

```
LSL     X9, X3, #3
ADD     X9, X6, X9
LDUR    X9, [X9, #0] // X9 = A[i]
LSL     X10, X4, #3
ADD     X10, X6, X10
LDUR    X10, [X10, #0] // X10 = A[j]
ADD     X9, X9, X10 // X9 = A[i] + A[j]
STUR    X9, [X7, #64] // B[8] = A[i] + A[j]
```

2.9

```
A[1] = &A[0]; f = 2*(&A[0]);
```

2.10

```
ADDI    X9, X6, #8 // I-type opcode=580, imm=8, Rn=6, Rd=9
ADD     X10, X6, XZR // R-type opcode=1112, Rm=31, Rn=6, Rd=10
STUR    X10, [X9, #0] // D-type opcode=1984, address=0, Rn=X9, Rt=X10
LDUR    X9, [X9, #0] // D-type opcode=1986, address=0, Rn=X9, Rt=X9
ADD     X0, X9, X10 // R-type opcode=1112, Rm=10, Rn=9, Rd=0
```

2.11.1 `0x5000000000000000`

2.11.2 Overflow

2.11.3 `0xB000000000000000`

2.11.4 No overflow assuming numbers are signed (two's complement)

2.11.5 `0xD000000000000000`

2.11.6 Overflow

2.13 R-type: `ADD X0, X0, X0`

2.14 D-type: `0xf8020149`

2.18.1 `0x1234567ababefef8`

2.18.2 `0x2345678123456780`

2.18.3 `0x0000000000000545`

2.20

```
EORI    X10, X11, #-1
```

2.21

```
LDUR    X14, [X13, #0]
LSL     X11, X14, #4
```

2.22 X1 = 2

2.25.1 The final value of X0 is 20.

2.25.2

```
acc = 0;
i = 10;
while (i > 0) {
    acc += 2;
    i--;
}
```

2.25.3 $5 \times N + 2$ instructions.

2.25.4 The final value of X0 is 22.

2.25.5 Note the change from > to >= in the **while** loop.

```
acc = 0;
i = 10;
while (i >= 0) {
    acc += 2;
    i--;
}
```

2.25.6 The **SUBIS** instruction sets the condition flag.

2.25.7

```
    SUBIS   X1, X1, #0
    B.LE    DONE
LOOP:
    SUBIS   X1, X1, #1
    ADDI    X0, X0, #2
    B.GT    LOOP
DONE:
```

2.26

```
    ORR     X10, XZR, XZR
LOOPI:
    SUBS    XZR, X10, X0
    B.GE    ENDI
    ORR     X11, XZR, XZR
LOOPJ:
```

```
        SUBS    XZR, X11, X1
        B.GE    ENDJ
        ADD     X12, X10, X11
        LSL     X13, X11, #5
        ADD     X14, X2, X13
        STUR    X12, [X14, #0]
        ADDI    X11, X11, #1
        B       LOOPJ
ENDJ:
        ADDI    X10, X10, #1
        B       LOOPI
ENDI:
```

2.27 153 instructions are being executed (answer depends on answer for 2.26).

2.28 Assuming X1 holds the base address of the integer array MemArray:

```
long long int i = 0;
long long int *p = MemArray;
do {
    result += *p;
    p++;
    i++;
}
while (i < 100);
```

2.29

```
        ADDI    X12, X1, #800
LOOP:
        LDUR    X11, [X1, #0]
        ADD     X0, X0, X11
        ADDI    X1, X1, #8
        CMP     X1, X12
        B.NE    LOOP
        ORR     X10, XZR, #100
```