Computer Architecture 1 - Übungsblatt 6

Dirk Heine, 2030941
Jan Hendrik Dithmar, 2031259
Steffen Heil, 2019358

Aufgabe 1

Because there is no further specification, we assume that we should write the program for the sequential machine.

```
**********************************************************************
* multiply
*
* input     R01 = first factor
*           R02 = second factor
*
* output    R01 is changed !
*           R02 is changed !
*           R03 is the product of R01 * R02 (if not failed)
*           R04 is 0 if succeeded, 1 if overflow
*
* limits    calculates only in range [-2^31 .. 2^31-1]
*           changes contents of R21 - R23
*
**********************************************************************


multiply:       add    R04 R00 R00          * optimistic: no error

* catch special cases

                bnez   R01 mul_anz          * CASE a == 0
                add    R03 R00 R00          * RETURN 0 / okay
                jr     R31                  *

mul_anz:        bnez   R02 mul_bnz          * CASE b == 0
                add    R03 R00 R00          * RETURN 0 / okay
                jr     R31                  *

mul_bnz:        addi   R22 R00 1            * R22 = 1

                xor    R21 R22 R01          *
                bnez   R21 mul_ano          * CASE a == 1
                add    R03 R02 R00          * RETURN b / okay
                jr     R31                  *

mul_ano:        xor    R21 R22 R02          *
                bnez   R21 mul_bno          * CASE b == 1
                add     R03 R01 R00          * RETURN a / okay
                jr      R31                  *

* force a,b positive

mul_bno:        slti    R21 R01 0            * R21 = a was negative
                slti    R22 R02 0            * R22 = b was negative

                beqz    R21 mul_apos         * a < 0 ?
                sub     R01 R00 R01          * make a positive

                                             * special case a = -2^31.
                                             * since b != 0, b != 1 fail.
                slti    R23 R01 0            * a < 0 ?
                bnez    R23 mul_error        * a was -2^31. cannot
                                             * multiply this.

mul_apos:       beqz    R22 mul_bpos         * b < 0 ?
                sub     R02 R00 R02          * make b positive

                                             * special case b = -2^31.
                                             * sine a != 0, a != 1 fail.
                slti    R23 R02 0            * b < 0 ?
                bnez    R23 mul_error        * b was -2^31. cannot
                                             * multiply this.

mul_bpos:       xor     R21 R21 R22          * R21 = result is negative
                sls     R22 R01 R02          * a < b ?
                beqz    R22 mul_noex         *

                xor     R01 R01 R02          * yes: exchange a <-> b
                xor     R02 R01 R02          *
                xor     R01 R01 R02          *     :-) for the fun - factor

* multiply
* assert: a >= b > 0,  loop:  a shl 1,  b shr 1,  break := (b == 0)

mul_noex:       addi    R03 R00 0            * R03 = result = 0

mul_loop:       andi    R23 R02 1            *
                beqz    R23 mul_shift        * is least significant bit
                                             * of R02 = 1?
                add     R03 R03 R01          * yes: R03 += R01
                slti   R23 R03 0            *      check for overflow
                bnez   R23 mul_error        *      yes: goto error

mul_shift:      slli   R01 R01              * a <<= 1
                srli   R02 R02              * b >>= 1
                beqz   R02 mul_finish       * are we finished ?
                slti   R23 R01 0            * overflow at a ?
                beqz   R23 mul_loop         * no: next step

* very special case: - 2^(x) * 2 ^ (31-x) = -2^31 !!!
* this case cannot be done by this algorithm
* but it shifts to following values:
* a = 0x80000000, b = 0x00000001, R3 still zero

                bnez   R03 mul_error        * nothing added so far ?
                add    R03 R01 R00          * copy a to result (in this
                                            * special case only)
                seqi   R02 R02 1            * really b == 0x00000001 ?
                beqz   R02 mul_error        *
                slli   R01 R01              *
                bnez   R01 mul_error        * really a == 0x80000000 ?
                beqz   R21 mul_error        * result be negative ?
                jr     R31                  * PROOFED: the special case.

mul_error:      addi   R04 R00 1            * overflow occured.
                jr     R31

mul_finish:     beqz   R21 mul_return       * result should be negative ?
                sub    R03 R00 R03

mul_return:     jr     R31                  * here we go. :-)
```

Aufgabe 3

1. $reset = 1$:
   Initially only stage $k = 0$ is full ($t = k$). $full^t.k \Leftrightarrow t \geq k$ holds. $\surd$

2. $reset = 0$:

Aufgabe 4

Two cases:

- $m = k$:

$$\begin{aligned} I(k,t) &= I(m,t) - (k-m) \\ &\overset{m=k}{=} I(k,t) - (k-k) \\ &= I(k,t) - 0 \\ &= I(k,t) \end{aligned}$$

- $m < k$:

$$\begin{aligned} I(k,t) &= I(m,t) - (k-m) \\ \Leftrightarrow I(m,t) &= I(k,t) + (k-m) \quad \text{Def. of } I \end{aligned}$$