Computer Architecture HW3

Tony G. Liu B05202068

November 29, 2020

1 ALU

- Signed Add: Overflow condition- `{a[MSB]+b[MSB], (a+b)[MSB]}== 2'b10, 2'b01`
- Signed Sub: Overflow condition- `{a[MSB]-b[MSB], (a-b)[MSB]}== 2'b01, 2'b10`
- Signed Mul: Overflow condition- `{a[MSB] ^ b[MSB], (a*b)[MSB]}== 2'b00, 2'b11`
- Signed Max/Min: Take the `$signed(..)` before comparing two numbers.
- Unsigned Add: Overflow condition- `{a[MSB]+b[MSB], (a+b)[MSB]}== 2'b00(a[MSB],b[MSB] == 1), 2'b10`
- Unsigned Sub: Overflow condition- `a < b`
- Unsigned Mul: Overflow condition- the sum of the leftmost set bit position of `a` and `b` is over 31
- Unsigned Max/Min: Simpley do the comparison
- And: `a & b`
- Or: `a | b`
- Xor: `a ^ b`
- BitFlip: `~a`
- BitReverse: Keep putting the set bits of `a` in the output until `a` becomes zero. After `a` becomes zero, shift the remaining bits of the output by the number of count. It takes $O(\log n)$ time complexity.

2 FPU

2.1 Add

1. Sort the two numbers `i_data_a` and `i_data_b` so that the first operand `a` is no less than the second operand `b`.
2. Do the number alignment:

- Get the exponent difference `exp_diff`
- Right shift the significand of `b` by the exponent difference and increment the exponent of `b` by the difference

3. `OR` the last `exp_diff` -1 bits of significand `b` and let it be `S`
4. Add significand `a` and the shifted significand `b` if their sign are the same, subtract if otherwise.
5. Check the carry bit of the significand `sig_add` after adding.
6. `R` would be the bit right after the LSB of `sig_add`, and `S` will be the `OR` of rest of the remaining bits.
7. Increment or decrement the amswer by 1(depending on the sign bits) if the bits `{R, S} == 2'b11, 2b'10`

2.2 Multiplication

1. Take the product of two significands of `i_data_a` and `i_data_b`.
2. `OR` the last 32 bits of the product and let it be `S`(= `sig_product[31:0]`)
3. Do the normalization(*i.e.*, check the MSB of the product)
4. Take the first 32 bits of the product and plus `sig_product[32]&S`
5. Add the two exponent and minus 127. Increment by 1 if the normalization bit is set.
6. Putting the sign bit, `sig_product` and the exponent result together.

3 CPU

The block diagram is mostly the same as the one in textbook, except that we have to wait for several cycles for the data in instruction memory and data memory to be ready.