Lab 08

R. Ferrero, A. C. Marceddu

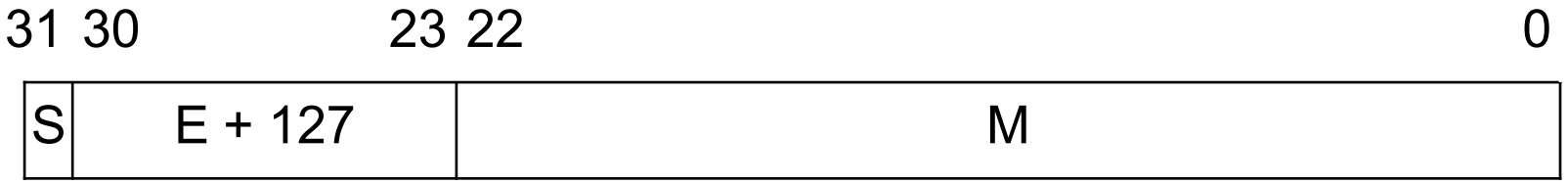
Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)

Torino - Italy

Representation of floating point

- The normalized representation of a floating point number is $\pm X * 2^E$, where $1 \leq X < 2$
- IEEE-754 SP standard expresses floating point numbers in 32 bits:
  - Sign S: 0 if positive, 1 if negative.
  - Exponent E with bias 127.
  - Mantissa M: fractional part of X.

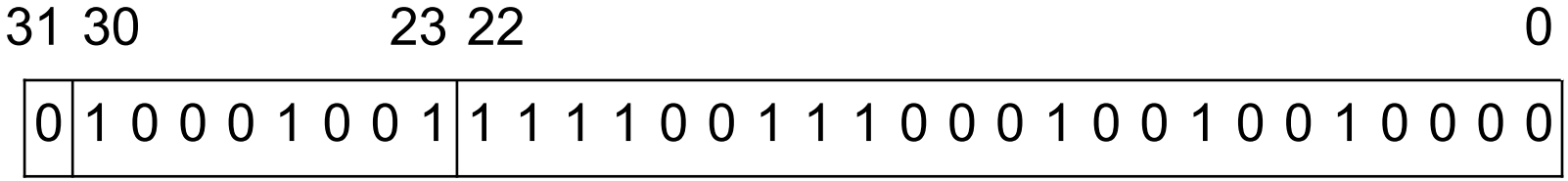


Exercise

- There is not any native implementation of floating point numbers in Cortex-M3: we want to implement it in software.
- If `Rn` contains the integer part and `Rm` the fractional part, write in `Rd` the floating point number according to IEEE-754 SP standard.

Example

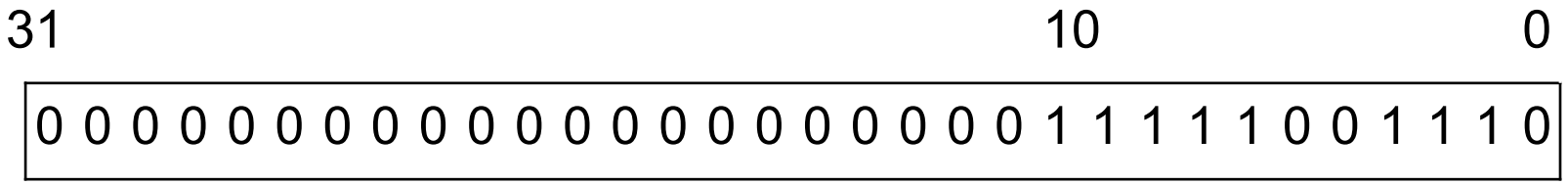
- `Rn` =1998, `Rm` =142578125
- 1998.142578125 expressed in normalized scientific notation is $1.9513111145 * 2^{10}$
- $E + 127 = 137 = 10001001_2$



- You can check the result here: https://www.h-schmidt.net/FloatConverter/IEEE754.html

Computation of the exponent

- The exponent is equal to the position of the first bit set to 1 in the binary representation of the integer part.
- Example: $1998 = 11111001110_2$



- After adding the bias (127), E = 137

Computation of the mantissa (1)

- The first bits of the mantissa are taken from the binary representation of the integer part, after removing the initial '1'.
- Example: 1998 = 11111001110_2
  - The first 10 bits of the mantissa are 1111001110
- The remaining N bits of the mantissa are obtained by converting the fractional part.
- In the conversion, the fractional part is interpreted as an integer number.
- Example: X = 142578125

Computation of the mantissa (2)

- Example: X = 142578125
- Let P be the lowest power of 10 which is higher than X. P = 1000000000
- The following loop is repeated N times:
  - X is doubled
  - if the result is higher than P
    - the next bit of the mantissa is 1 and the new value of X is X = X – P
  - else the next bit of the mantissa is 0
  - repeat loop

Computation of the mantissa (3)

iteration	X	2 * X	bit
1	142578125	285156250	0
2	285156250	570312500	0
3	570312500	1140625000	1
4	140625000	281250000	0
5	281250000	562500000	0
6	562500000	1125000000	1
7	125000000	250000000	0
8	250000000	500000000	0
9	500000000	1000000000	1
10	0	0	0
11	0	0	0
12	0	0	0
13	0	0	0

Conversion to IEEE-754 SP

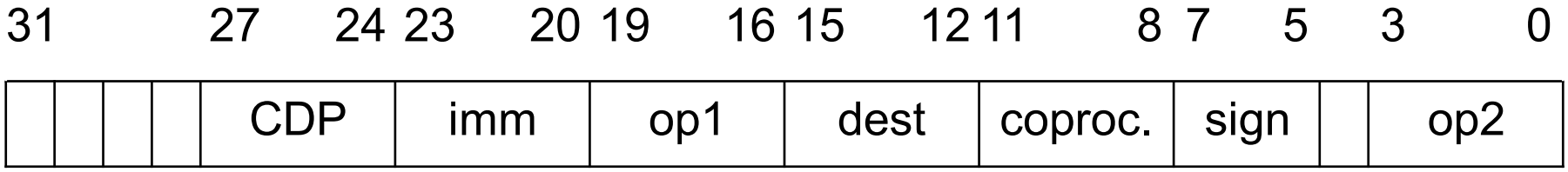
We call a coprocessor as follows:

`CDP proc, imm, dest, op1, op2, sign`

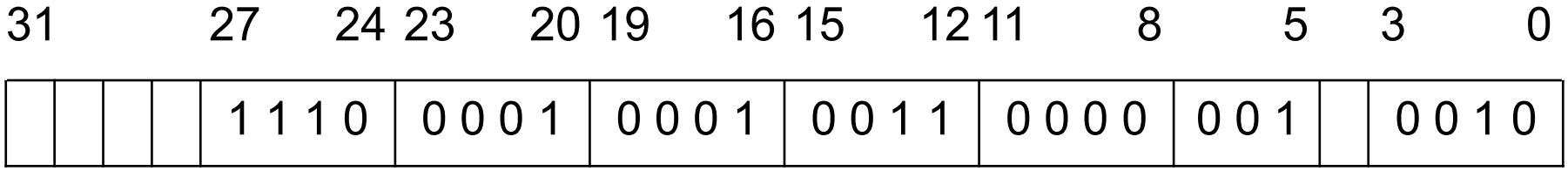
- `CDP`: instruction to call a coprocessor.
- `proc`: called coprocessor. We call `p0`.
- `imm`: operation executed by coprocessor.
  - `imm` = 1 for conversion to the IEEE-754 SP.
- `dest`, `op1`, `op2`: registers containing result, integer part, fractional part, respectively.
- `sign`: 0 if positive, 1 if negative.

CDP encoding

- Only meaningful bits are shown:



- The registers used by a coprocessor are named `c0`, `c1`, ..., `c15`.
- Example: `CDP p0, #1, c3, c1, c2, #1`



Software implementation

- Cortex-M3 has not any coprocessor.
- `CDP` raises a usage fault.
  - usage fault must be enabled, otherwise a hard fault is raised.
- The conversion to the IEEE-754 SP format is done in the exception handler.
- The following slides list the steps to be implemented in the exception handler.

1) Recognizing coprocessor fault

- Check the proper bit in the Usage Fault Status Register.
- If the exception is due to a coprocessor instruction, branch to the corresponding piece of code.
- Otherwise, write a dummy implementation of other usage faults (e.g., `B .`).

2) Recognizing offending instruction

- Before entering the exception handler, `PC` is saved in the stack (`MSP` or `PSP`) with offset 24.
  - use of `MSP` or `PSP` is determined by reading `LR`.
- Load 4 bytes (1 word) from the address of PC
  - due to little endianness, the two halfwords must be switched (e.g., by rotating 16 positions).
- If the instruction is `CDP p0, #1, ...` execute code for IEEE-754 SP conversion.
  - bits are xxxx 1110 0001 xxxx xxxx 0000 xxxx xxxx

3) Changing return address

- Usage faults return to the same instructions that triggered the fault.
- Since we do not want to execute `CDP` again after the handler, the return address must be updated to the next instruction.
- Update the value of `PC` saved in the stack (with offset 24) with the new value `PC` + 4.

4) Accessing source registers

- Index of registers ranges between 0 and 7.
- We assume that `r4` corresponds to `c0`, `r5` = `c1`, `r6` = `c2`, ..., `r11` = `c7`.
- Save registers `r4-r11` in the stack (as required by AAPCS).
  - instructions in steps 1, 2, 3 can not modify `r4-r11`.
- After extracting the index of a source register from the encoded instruction, you can extract its content from the stack with offset index * 4.

5) Computing the result

- Bit 5 in the encoded instruction (representing the sign) is the most significant bit of the result.
- The other bits are set by computing exponent and mantissa.
- Finally, update the destination register saved in the stack with the new computed value.

6) Concluding the handler

- Restore values of register `r4-r11` with a `POP`.
  - The destination register will contains the IEEE-754 SP representation, since the corresponding entry in the stack was updated in step 5.
- Return to the program with `BX LR`.
  - the next instruction will be the one after `CDP`, since the value of `PC` (automatically retrieved from the stack) has been updated in step 3.