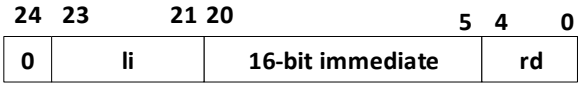
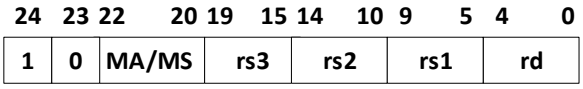
The type of instructions are given below:

1. Load immediate instruction format



li: Load a 16-bit immediate value from the [20:5] instruction field into the 16-bit field specified by the li field [23-21] of the 128-bit register rd.

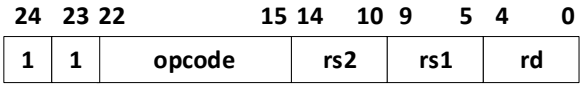
2. Multiply-add and multiply-subtract (low/high) R4-instruction format



In the table below, 32-bit signed integer add and subtract operations are performed with '**saturate to signed word**' rounding that takes a 32-bit signed integer result, and converts it to -2^{31} if it's less than -2^{31}, to $+2^{31}-1$ if it's greater than $+2^{31}-1$, and leaves it unchanged otherwise. Multiply operations on 16-bit signed operands calculate 32-bit integer signed products.

MA/MS/l/h 22-20	**Description of instruction code [22-20]**
x00	**Signed integer multiple-add low with saturation:** Multiply low 16-bit-fields of each 32-bit field of registers **rs3** and **rs2,** then add 32-bit products to 32-bit fields of register **rs1,** and save result in register **rd**.
x01	**Signed integer multiple-add high with saturation:** Multiply high 16-bit-fields of each 32-bit field of registers **rs3** and **rs2,** then add 32-bit products to 32-bit fields of register **rs1,** and save result in register **rd**.
x10	**Signed integer multiple-subtract low with saturation:** Multiply low 16-bit-fields of each 32-bit field of registers **rs3** and **rs2,** then subtract 32-bit products from 32-bit fields of register **rs1,** and save result in register **rd**.
x11	**Signed integer multiple-subtract high with saturation:** Multiply high 16-bit-fields of each 32-bit field of registers **rs3** and **rs2,** then subtract 32-bit products from 32-bit fields of register **rs1,** and save result in register **rd**.

3. R3-instruction format



In the table below, 16-bit signed integer add (**ahs**) and subtract (**sfhs**) operations are performed with **'saturate to signed word'** rounding that takes a 16-bit signed integer X, and converts it to −32768 if it's less than −32768, to +32767 if it's greater than 32767, and leaves it unchanged otherwise.

Opcode 22-15	**Description of Instruction Opcode**
xxxx0000	**Nop**
xxxx0001	**bcw:** broadcast a right 32-bit word of register **rs1** to each of the four 32-bit words of register **rd.**
xxxx0010	**and**: bitwise ***logical and*** of the contents of registers ***rs1*** and ***rs2***
xxxx0011	**or**: bitwise ***logical or*** of the contents of registers ***rs1*** and ***rs2***
xxxx0100	**popcnth**: *count ones in halfwords*: the number of 1s in each of the four halfword-slots in register ***rs1*** is computed. If the halfword slot in register ***rs1*** is zero, the result is 0. Each of the results is placed into corresponding 16-bit slot in register ***rd***. ***(Comments: 8 separate 16-bit halfword values in each 128-bit register)***
xxxx0101	**clz**: *count leading zeroes in words*: for each of the two 32-bit word slots in register ***rs1*** the number of zero bits to the left of the first non-zero bit is computed. If the word slot in register ***rs1*** is zero, the result is 32. The two results are placed into the corresponding 32-bit word slots in register ***rd***. ***(Comments: 4 separate 32-bit values in each 128-bit register)***
xxxx0110	**rot:** *rotate right:* the contents of register ***rs1*** are rotated to the right according to the count in the 7 least significant bits (6 to 0) of the contents of register ***rs2***. The result is placed in register ***rd***. If the count is zero, the contents of register ***rs1*** are copied unchanged into register ***rd***. Bits rotated out of the right end of the 128-bit contents of register ***rs1*** are rotated in at the left end.
xxxx0111	**shlhi**: *shift left halfword immediate*: packed 16-bit halfword shift left logical of the contents of register ***rs1*** by the 4-bit immediate value of instruction field ***rs2***. Each of the results is placed into the corresponding 16-bit slot in register ***rd***. ***(Comments: 8 separate 16-bit values in each 128-bit register)***

xxxx1000	**a**: *add word*: packed 32-bit unsigned add of the contents of registers ***rs1*** and ***rs2*** ***(Comments: 4 separate 32-bit values in each 128-bit register)***
xxxx1001	**sfw**: *subtract from word*: (packed) 32-bit unsigned subtract of the contents of registers ***rs1*** and ***rs2 (Comments: 4 separate 32-bit values in each 128-bit register)***
xxxx1010	**ah**: *add halfword* : (packed) (16-bit) halfword unsigned add of the contents of registers ***rs1*** and ***rs2 (Comments: 8 separate 16-bit values in each128-bit register)***
xxxx1011	**sfh**: *subtract from halfword*: (packed) (16-bit) halfword unsigned subtract of the contents of registers ***rs1*** and ***rs2. (Comments: 8 separate 16-bit values in each128-bit register)***
xxxx1100	**ahs**: *add halfword saturated*: (packed) (16-bit) halfword signed add **with saturation** of the contents of registers ***rs1*** and ***rs2. (Comments: 8 separate 16-bit values in each128-bit register)***
xxxx1101	**sfhs**: *subtract from halfword saturated*: (packed) (16-bit) signed subtract **with saturation** of the contents of registers ***rs1*** and ***rs2. (Comments: 8 separate 16-bit values in each128-bit register)***
xxxx1110	**mpyu**: *multiply unsigned*: the 16 rightmost bits of each of the four 32-bit slots in registers ***rs1*** are multiplied by the 16 rightmost bits of the corresponding 32-bit slots in register ***rs2***, treating both operands as unsigned. The four 32-bit products are placed into the corresponding slots of register ***rd***. ***(Comments: 4 separate 32-bit values in each 128-bit register)***
xxxx1111	**absdb**: *absolute difference of bytes*: the contents of each of the 16 byte slots in register ***rs2*** is subtracted from the contents of the corresponding byte slot in register ***rs1***. The absolute value of each of the results is placed into the corresponding byte slot in register ***rd***. ***(Comments: 16 separate 8-bit values in each 128-bit register)***

The basic structure of the multimedia unit is provided below: