支持按字节访存

为支持 `lb, lh, sb, sh` 等指令，我们需新增数据扩展模块，以按字节来访问内存。

字节使能

`m_data_byteen[3:0]` 是字节使能信号，其最高位到最低位分别与 `m_data_wdata` 的 `[31:24]`、`[23:16]`、`[15:8]` 及 `[7:0]` 对应（即一位对应一个字节）。

例如，若 `m_data_byteen[3]` 为 1，则 `m_data_wdata[31:24]` 会被写入到 `m_data_addr` 所指向 word 的 `[31:24]`，依此类推。

若 `m_data_byteen` 的任意一位为 1，则代表当前需要写入内存，也就是说可以用 `|m_data_byteen` 代替数据存储器的写使能信号。

`m_data_byteen[3:0]` 主要用于支持 `sb`、`sh` 这两条指令。当处理器执行 `sb`、`sh` 指令时，只要对 EX/MEM 保存的 ALU 计算结果即 32 位地址的低两位进行解读，产生相应的 `m_data_byteen` 信号，就可以“通知” Testbench 中的 DM 该写入哪些字节，具体规则如下：

sw 指令：向内存写入对应的字

地址[1:0]	m_data_byteen [3:0]	用途
XX	1111	`m_data_wdata[31:24]` 写入 byte3 `m_data_wdata[23:16]` 写入 byte2 `m_data_wdata[15:8]` 写入 byte1 `m_data_wdata[7:0]` 写入 byte0

sh 指令：向内存写入对应的半字

地址[1:0]	m_data_byteen [3:0]	用途
0X	0011	`m_data_wdata[15:8]` 写入 byte1 `m_data_wdata[7:0]` 写入 byte0
1X	1100	`m_data_wdata[31:24]` 写入 byte3 `m_data_wdata[23:16]` 写入 byte2

sb 指令：向内存写入对应的字节

地址[1:0]	m_data_byteen [3:0]	用途
00	0001	`m_data_wdata[7:0]` 写入 byte0
01	0010	`m_data_wdata[15:8]` 写入 byte1
10	0100	`m_data_wdata[23:16]` 写入 byte2
11	1000	`m_data_wdata[31:24]` 写入 byte3

BE 扩展模块需放在与数据存储器交互数据之前。显然，BE 扩展功能部件还需要有来自控制器的控制信号。

数据扩展模块

对于 `lb`、`lh` 来说，我们需要额外增加一个数据扩展模块。这个模块把从数据存储器读出的数据做符号扩展。

以 `lb` 为例，数据扩展模块输入数据寄存器的 32 位数据，根据 ALU 计算出来的地址最低 2 位从中取出**特定的字节**，并以该字节的最高位为符号位做符号扩展。

参考的接口定义如下：

信号名	方向	描述
A[1:0]	I	最低两位的地址
Din[31:0]	I	输入的 32 位数据
Op[2:0]	I	数据扩展控制码 000：无扩展 001：无符号字节数据扩展 010：符号字节数据扩展 011：无符号半字数据扩展 100：符号半字数据扩展
Dout[31:0]	O	扩展后的 32 位数据

思考题

请问采用字节使能信号的方式处理写指令有什么好处？（提示：从清晰性、统一性等角度考虑）

思考题

请思考，我们在按字节读和按字节写时，实际从 DM 获得的数据和向 DM 写入的数据是否是一字节？在什么情况下我们按字节读和按字节写的效率会高于按字读和按字写呢？