数据存取指令相关部件设计文档

1. 需求分析
2. 指令行为描述

所有指令均需要计算地址：base寄存器加上符号扩展的offset，也即均需要经过ALU。

最小操作对象为1个字节。

|  |  |  |
| --- | --- | --- |
| 指令 | 格式 | 功能 |
| LB | lb rt,offset(base) | 从目标地址读取一个字节，符号扩展后存入rt |
| LBU | lbu rt,offset(base) | 从目标地址读取一个字节，逻辑扩展后存入rt |
| LH | lh rt,offset(base) | 从目标地址读取一个半字，符号扩展后存入rt |
| LHU | lhu rt,offset(base) | 从目标地址读取一个半字，逻辑扩展后存入rt |
| LW | lw rt,offset(base) | 从目标地址读取32位数据并存入rt |
| LWL | lwl rt,offset(base) | 从目标地址读取1~4个字节（对应地址末两位的0~3），从存储器低位取，从寄存器高位存 |
| LWR | lwr rt,offset(base) | 从目标地址读取4~1个字节（对应地址末两位的0~3），从存储器高位取，从寄存器低位存 |
| SB | sb rt,offset(base) | 向目标地址写入rt的最低字节 |
| SH | sh rt,offset(base) | 向目标地址写入rt的低半字 |
| SW | sw rt,offset(base) | 向目标地址写入rt的内容 |
| SWL | swl rt,offset(base) | 向目标地址写入1~4个字节（对应地址末两位的0~3），从rt的高位取，向地址的低位存 |
| SWR | swr rt,offset(base) | 向目标地址写入4~1个字节（对应地址末两位的0~3），从rt的低位取，向地址的高位存 |

1. 部件功能

从上文的指令描述中不难看出我们需要两类部件：位写使能生成器和数据生成器。

1. 位写使能生成器

不同的load和store指令涉及的数据位数是不同的，由地址末两位确定（小端方式），load需要控制寄存器的位写使能，store需要控制存储器的位写使能。

Load指令和寄存器位写使能对应关系

|  |  |  |
| --- | --- | --- |
| 指令 | addr[1:0] | reg\_byte\_write\_en |
| lwl | 0 | 4’b1111 |
| 1 | 4’b1110 |
| 2 | 4’b1100 |
| 3 | 4’b1000 |
| lwr | 0 | 4’b0001 |
| 1 | 4’b0011 |
| 2 | 4’b0111 |
| 3 | 4’b1111 |
| 剩余所有load指令 | xx | 4’b1111 |

Store指令和存储器写使能对应关系

|  |  |  |
| --- | --- | --- |
| 指令 | addr[1:0] | mem\_byte\_write\_en |
| sb | 0 | 4’b1000 |
| 1 | 4’b0100 |
| 2 | 4’b0010 |
| 3 | 4’b0001 |
| sh | 0 | 4’b1100 |
| 2 | 4’b0011 |
| sw | xx | 4‘b1111 |
| swl | 0 | 4‘b1111 |
| 1 | 4‘b0111 |
| 2 | 4‘b0011 |
| 3 | 4‘b0001 |
| swr | 0 | 4‘b1000 |
| 1 | 4‘b1100 |
| 2 | 4‘b1110 |
| 3 | 4‘b1111 |

注：上述部件位于EXE段

1. 数据生成器

对于load指令而言，从存储器取出数据之后需要进行位扩展（lb，lbu，lh，lhu）或移位（lwl，lwr），对于store指令而言（swl，swr），需要存储的数据也需要进行移位，这些操作都需要一个单独的部件来处理。

需要注意的是，store数据的处理应位于EXE段，load数据的处理应位于MEM段。

1. 输入输出信号
2. 位写使能生成器

Load指令的位写使能生成器

|  |  |  |
| --- | --- | --- |
| 输入 | addr[1:0] | 地址末两位 |
| load\_sel[2:0] | 用于区分load指令 |
| 输出 | b\_w\_en[3:0] | 位写使能 |

Load指令和load\_sel的对应关系如下：

|  |  |
| --- | --- |
| 指令 | load\_sel |
| LB | 0 |
| LBU | 1 |
| LH | 2 |
| LHU | 3 |
| LW | 4 |
| LWL | 5 |
| LWR | 6 |

Store指令的位写使能生成器

|  |  |  |
| --- | --- | --- |
| 输入 | addr[1:0] | 地址末两位 |
| store\_sel[2:0] | 用于区分store指令 |
| 输出 | b\_w\_en[3:0] | 位写使能 |

Store指令与load\_sel的对应关系如下：

|  |  |
| --- | --- |
| 指令 | store\_sel |
| SB | 0 |
| SH | 1 |
| SW | 2 |
| SWL | 3 |
| SWR | 4 |

1. 数据生成器

Load指令的数据生成器

|  |  |  |
| --- | --- | --- |
| 输入 | addr[1:0] | 地址末两位 |
| load\_sel[2:0] | 用于区分load指令 |
| mem\_data[31:0] | 来自存储器的数据 |
| 输出 | data\_to\_reg[31:0] | 处理后的数据 |

Store指令的数据生成器

|  |  |  |
| --- | --- | --- |
| 输入 | addr[1:0] | 地址末两位 |
| store\_sel[2:0] | 用于区分store指令 |
| rt\_data[31:0] | 来自寄存器的数据 |
| 输出 | real\_rt\_data[31:0] | 处理后的数据 |

1. 实现思路
2. 位写使能生成

这一过程可以直接根据上文的指令和信号对应关系表得出，为不同的信号提供对应的位写使能信号即可。

1. 数据生成
2. Load相关

根据不同的指令对数据进行相应的操作，lb、lh需要符号扩展，lbu、lhu需要逻辑扩展，lwl需要对数据进行左移，lwr需要对数据进行右移。

同时根据地址末两位的不同，取出的数据也有所不同：

|  |  |  |
| --- | --- | --- |
| 指令 | 地址末两位 | 所取数据(mem\_data） |
| lb/lbu | 00 | mem\_data[31:24] |
| 01 | mem\_data[23:16] |
| 10 | mem\_data[15:8] |
| 11 | mem\_data[7:0] |
| lh/lhu | 00 | mem\_data[31:16] |
| 10 | mem\_data[15:0] |
| lw | xx | mem\_data |
| lwl | 00 | mem\_data |
| 01 | mem\_data<<8 |
| 10 | mem\_data<<16 |
| 11 | mem\_data<<24 |
| lwr | 00 | mem\_data |
| 01 | mem\_data>>8 |
| 10 | mem\_data>>16 |
| 11 | mem\_data>>24 |

（注：采用大端模式，低地址存高字节）

1. Store相关

同样根据不同的指令对数据进行相应的操作，swl需要对数据进行左移，swr需要对数据进行右移。

根据地址末两位的不同，虽然要存的数据相同，但是存储的位置不同：

|  |  |  |
| --- | --- | --- |
| 指令 | 地址末两位 | 所存数据(rt\_data) |
| sb | 00 | rt\_data<<24 |
| 01 | rt\_data<<16 |
| 10 | rt\_data<<8 |
| 11 | rt\_data |
| sh | 00 | rt\_data<<16 |
| 10 | rt\_data |
| sw | xx | rt\_data |
| swl | 00 | rt\_data |
| 01 | rt\_data>>8 |
| 10 | rt\_data>>16 |
| 11 | rt\_data>>24 |
| swr | 00 | rt\_data<<24 |
| 01 | rt\_data<<16 |
| 10 | rt\_data<<8 |
| 11 | rt\_data |