6.823 Computer System Architecture

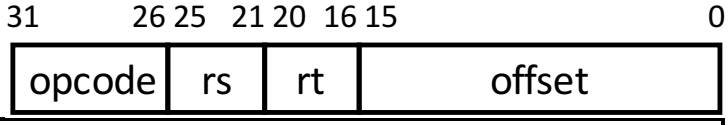
BigMIPS ISA

http://csg.csail.mit.edu/6.823/

In most modern processors, general-purpose registers are used to hold both data and addresses. This ties the width of each general-purpose register and the size of the address space (i.e., the maximum amount of addressable memory). For example, machines with 32-bit registers can address at most 2^{32} = 4GB of memory, which these days is too small for many applications. Machines with 64-bit registers can address 2^{64} = 16 EB (exabytes) of memory, which is plenty. But they use wider registers and ALUs, which consume more area, more power, and are slower.

To solve this problem, we define BigMIPS, a new ISA that supports 64-bit addresses but uses 32-bit registers (and therefore can be implemented with efficient 32-bit datapaths). BigMIPS modifies the load and store instructions of the MIPS ISA to use 64-bit addresses. Instructions other than Load Word and Store Word are identical to MIPS. Refer to the handout "RISC ISA – MIPS32" for the MIPS ISA.

BigMIPS loads and stores have exactly the same instruction format as MIPS loads and stores (I-type, i.e., *rs*, *rt*, and 16-bit *offset*). They build a 64-bit address by concatenating the contents of two consecutive 32-bit registers, *rs* and *rs+1*. Specifically, the effective address is the 64-bit sum of the 64-bit sign-extended *offset* and the contents of a 64-bit *base* value whose upper 32 bits are the contents of register *rs*, and its lower 32 bits are the contents of register *rs+1*. The value *rs* must be even, so that *rs+1* can be efficiently computed by *rs* OR 1. Table 1 gives the precise description of loads and stores.



Instruction	Format and Description

Load Word	`LW rt, offset(rs)`
	
	`rt ← Mem[((rs) << 32 | (rs+1)) + SignExt64(offset)]`
	
	Sign-extend *offset* and add to the concatenation of contents of registers *rs* and *rs+1* to form a 64-bit effective address. *rs* must be even. Load contents of the addressed word into register *rt*.

Store Word	`SW rt, offset(rs)`
	
	`Mem[((rs) << 32 | (rs+1)) + SignExt64(offset)] ← rt`
	
	Sign-extend *offset* and add to the concatenation of contents of registers *rs* and *rs+1* to form a 64-bit effective address. *rs* must be even. Store the contents of register *rt* at the addressed location.

Table 1. BigMIPS 64-bit-addressed LW and SW instructions.

For example, the following instructions load the word at address 0x0100000008 into register *R1*.

```
ADDI  R4, R0, 1  ;; R4 <- 1
ADDI  R5, R0, 0  ;; R5 <- 0
LW R1, 8(R4)
```

The following instruction is forbidden, because *rs* is odd:

```
SW R2, 8(R3)
```