Some Assembler Programs

This text is not meant to replace classes on programming. In Saarbrücken, where we teach, a class focusing on programming in MIPS assembly and C is presently taught in parallel with our system architecture class. The main purpose of this small chapter is to present programs for multiplication and division of binary numbers and of two’s complement numbers. These programs are later used in the compiler that we construct, as the code generated for the corresponding operations in the source language. We go over the material of this chapter in the classroom very quickly. We do however present the correctness proof for the school method of division from Sect. 9.3.1.

Before we present example programs, we have to say a few words about how we write assembler programs.

Assembler Syntax

An assembler program is a sequence of assembler instructions. The syntax of assembler instructions is given by the instruction-set-architecture tables in Sect. 8.1. The assembler translates instructions into machine words according to the MIPS instruction layout (see Fig. 55) in the straightforward way.

Representation of Constants in Assembly Language

We represent register numbers and immediate constants in decimal notation. Conversion to two’s complement representation is done by the assembler. Additionally, our assembler syntax allows us to specify immediate constants $i[15:0]$ and instruction indices $ii[25:0]$ given in binary representation by prepending 0b:

$$0b[i[15:0] \quad 0b[ii[25:0]]$$

1 In a hardware design lab the students can also have the experience that transferring the processor design of Chap. 8 literally to an FPGA board results in ... running hardware.

2 Note that most assemblers reference register $i$ by $\$i$. We omit the dollar sign in subsequent assembly programs for simplicity.
Other Conventions

We put comments behind a double slash ‘//’. Sometimes we number lines. These numbers are comments too; they just serve to compute jump distances.

9.1 Simple MIPS Programs

We present a few simple MIPS programs to be used in implementing assembler programs for software multiplication and division.

Storing a 32-bit Constant in a Register

Let $S[31:0] \in \mathbb{B}^{32}$. The following program loads this constant in register $k$ using two instructions. First, it stores the upper half and then the zero-extended lower half is ORed with the register content:

```
lui k 0bS[31:16]
or k k 0bS[15:0]
```

We abbreviate this program with $\text{gpr}(k) = S$.

Computing Sign and Absolute Value of an Integer

Assume a two’s complement number is stored in GPR register $i$. We want to store the sign and absolute value of this number in registers $j$ and $k$. That is, after executing the program starting from configuration $c$, we reach a configuration $c'$ with

$$c'.\text{gpr}(j) = \begin{cases} 1, & [c.\text{gpr}(i)] < 0, \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad \langle c'.\text{gpr}(k) \rangle = |[c.\text{gpr}(i)]|.$$  

For the sign we simply compare register $i$ with zero:

```
0: slti j i 0
```

To compute the absolute value, we invert and increment in case the number is negative. We invert by XORing with the mask $1^{32}$ which is obtained by a NOR of the mask $0^{32}$ with itself. In case the number is positive, we simply assign it to register $k$:

```
1: blez j 5 // if j=0 continue after line 5
2: nor k k k // generate mask
3: xor k k i // negate
4: addi k k 1
5: blez 0 2 // gpr(0) is always zero
6: addi k i 0
```

We abbreviate this program by $\text{sign-and-abs}(j,k,i)$. 