Lab Assignment: Computer Organization and Architecture Laboratory

Date: 22/10/21

Class	Instruction	Usage	Meaning
Arithmetic	Add	add rs,rt	$rs \leftarrow (rs) + (rt)$
	Comp	comp rs,rt	$rs \leftarrow$ 2's Complement (rs)
	Add immediate	addi rs,imm	$rs \leftarrow (rs) + imm$
	Complement Immediate	compi rs,imm	$rs \leftarrow$ 2's Complement (imm)
Logic	AND	and rs,rt	$rs \leftarrow (rs) \wedge (rt)$
	XOR	xor rs,rt	$rs \leftarrow (rs) \oplus (rt)$
Shift	Shift left logical	shll rs, sh	$rs \leftarrow (rs)$ left-shifted by sh
	Shift right logical	shrl rs, sh	$rs \leftarrow (rs)$ right-shifted by sh
	Shift left logical variable	shllv rs, rt	$rs \leftarrow (rs)$ left-shifted by (rt)
	Shift right logical	shrl rs, rt	$rs \leftarrow (rs)$ right-shifted by (rt)
	Shift right arithmetic	shra rs, sh	$rs \leftarrow (rs)$ arithmetic right-shifted by sh
	Shift right arithmetic variable	shrav rs, rt	$rs \leftarrow (rs)$ right-shifted by (rt)
Memory	Load Word	lw rt,imm(rs)	$rt \leftarrow mem[(rs) + imm]$
	Store Word	sw rt,imm,(rs)	$mem[(rs) + imm] \leftarrow (rt)$
Branch	Unconditional branch	b L	goto L
	Branch Register	br rs	goto (rs)
	Branch on less than 0	bltz rs,L	if$(rs) < 0$ then goto L
	Branch on flag zero	bz rs,L	if $(rs) = 0$ then goto L
	Branch on flag not zero	bnz rs,L	if$(rs) \neq 0$ then goto L
	Branch and link	bl L	goto L; 31 $\leftarrow$ (PC)+4
	Branch on Carry	bcy L	goto L if Carry = 1
	Branch on No Carry	bncy L	goto L if Carry = 0

Our processor KGP-RISC has the above Instruction Set Architecture (ISA). Assume that the processor has a 32 bit word, with all the registers and memory elements having 32 bit data. The address line of the memory is also 32 bits.

We are to develop first the op-code format for the above instruction set, identify the data path element and design the data path along with the control signals. Subsequently, we shall first develop a single-cycle instruction execution unit for KGP-RISC.

Proceed step-by-step as follows:

1. For the above Instruction set, evolve a suitable instruction format. Clearly specify the fields of the opcode and mention how each of the above instructions are to be encoded. Keep in mind, while deciding the op-code, you should keep provisions for adding more instructions to the ISA.

2. Identify the Data path elements and draw an architecture for the Arithmetic Logic Unit. Clearly differentiate between the data lines and control lines in your architecture diagram.

3. Draw the complete data path along with the control signals for a single cycle execution unit for the above ISA.

4. Write the truth table for the controller signals as a function of the opcode and the function code in the instruction format. Note that for a single cycle implementation of the controller there is no state and the design is purely combinational.

5. Develop the complete RTL Verilog code for the processor. Your design should have separate and modular codes for the data-path and control-path.

   As an indicative guideline, your submission should consist of the following verilog files:

   (a) KGPRISC.v
   (b) datapath.v
   (c) controller.v
   (d) registerfile.v
   (e) ALU.v
   (f) InstructionMemory.v
   (g) DataMemory.v

   Note these are indicative names of the modules, and you can make alterations as you deem necessary. You can also add to it depending on the architecture you need for supporting the ISA of KGPRISC.

6. Please write suitable test-benches for verifying these components, either individually or few modules together. What is mandatory is the top-level testbench file. Your testbench should correspond to an assembly level program for your favorite algorithm. For example, you can choose to implement Computegcd, or SortNumbers, etc. The input to these algorithms can be numbers stored in the data-memory, and accessed by appropriate instructions by your processor.