Texas Tech University
Department of Computer Science

Course Name: Computer Organization and Assembly Language Programming
Number: CS3350
Instructor: Dr. Juan Carlos Rojas **Email:** Juan-Carlos.Rojas@ttu.edu

TUCA-5.1 Computer Architecture Specification

This document describes the TTU Computer Architecture, version 5.1 (TUCA-5.1).

General Description

The TUCA-5.1 architecture is an 8-bit machine that follows a GPR load-store architecture. It has 16 general purpose registers labeled r0 to r15, an 8-bit ALU, and a 256-byte programmable 8-bit data memory with addresses from 0 to 0xFF.

It also has an instruction memory of up to 4096 bytes, with a fixed-length encoding of 2 bytes per instruction. See Instruction Encoding below for details.

Instruction Set

The instruction set supports the following instructions:

Load/Store Instructions

Instruction	Description
ld addr reg	Loads the 8-bit value from memory at location *addr* into the register *reg*.
ldr reg1 reg2	Loads the 8-bit value from memory at a location stored in *reg1*, into the register *reg2*.
ldi val reg	Loads the 8-bit value *val* into the register *reg*.
st reg addr	Stores the 8-bit value on register *reg* into the memory at location *addr*.
str reg1 reg2	Stores the 8-bit value on *reg1* into the memory at a location stored in *reg2*.

Arithmetic & Logic Instructions

Instruction	Description
add reg1 reg2 reg3	Adds the values in *reg1* and *reg2* and puts the result on *reg3*.
and reg1 reg2 reg3	Performs bitwise AND between *reg1* and *reg2* and puts the result on *reg3*.
or reg1 reg2 reg3	Performs bitwise OR between *reg1* and *reg2* and puts the result on *reg3*.
not reg1 reg2	Performs bitwise NOT on *reg1* and puts the result in *reg2*.
neg reg1 reg2	Negates the value in *reg1* using two's complement and puts the result in *reg2*.

Bit Manipulation Instructions

Instruction	Description
shl reg1 n reg2	Left-shifts the value on *reg1* by *n* bits and puts the result on *reg2*. n is a value from 1 to 7.
shr reg1 n reg2	Right-shifts the value on *reg1* by *n* bits and puts the result on *reg2*. n is a value from 1 to 7.

Comparison and Conditional Execution Instructions

Instruction	Description
eq reg1 reg2 reg3	Puts a 1 in *reg3* if the value of *reg1* is equal to that of *reg2*. Otherwise puts 0 into *reg3*.
gt reg1 reg2 reg3	Puts a 1 in *reg3* if the value of *reg1* is greater than that of *reg2*. Otherwise puts 0 into *reg3*. Note that this instruction interprets all values as unsigned.

if reg1	Execute the next instruction if the value of *reg1* is non-zero. Otherwise skip it.
skipif reg1	Skip the next instruction if the value of *reg1* is non-zero.

Branch and Program Control Instructions

Instruction	Description
jmp addr	Jumps and continues execution starting with the instruction stored at location *addr* of the instruction memory. Note that instruction addresses are generally not known in the assembly programs. Symbolic labels can be used. See the Assembly Language reference manual for details.
jmpr reg_hi reg_lo	Jumps and continues execution starting with instructions stored at a location of instruction memory stored in *reg_hi & reg_lo*. Instruction addresses are 12 bits. The 4 most significant bits of the address will be read from *reg_hi*, and the 8 least significant bits will be read from *reg_lo*.
readpc reg_hi reg_lo	Saves the contents of the current program counter (PC) into registers *reg_hi & reg_lo*. The PC is a 12-bit address, whose most significant 4 bits will be stored in *reg_hi*, and the least significant 8 bits will be stored in *reg_lo*.
halt	Terminates the program.

Instruction Encoding

The following instruction encoding is envisioned for this architecture. In practice it is not used for programming or simulation.

Instruction	Nibble 1 (4 bits)	Nibble 2 (4 bits)	Nibble 3 (4 bits)	Nibble 4 (4 bits)
jmp addr	0000	address		
ld addr reg	0001	address		reg

ldi reg val	0010	value		reg
st reg addr	0011	reg	address	
add reg1 reg2 reg3	0100	reg1	reg2	reg3
and reg1 reg2 reg3	0101	reg1	reg2	reg3
or reg1 reg2 reg3	0110	reg1	reg2	reg3
not reg1 reg2	0111	reg1	reg2	
neg reg1 reg2	1000	reg1	reg2	
shl reg1 n reg2	1001	reg1	n	reg2
shr reg1 n reg2	1010	reg1	n	reg2
eq reg1 reg2 reg3	1011	reg1	reg2	reg3
gt reg1 reg2 reg3	1100	reg1	reg2	reg3
if reg1	1101	reg1		
skipif reg1	1110	reg1		
halt	1111	0000		
ldr reg1 reg2	1111	0001	reg1	reg2
str reg1 reg2	1111	0010	reg1	reg2
jmpr reg1 reg2	1111	0011	reg1	reg2
loadpc reg1 reg2	1111	0100	reg1	reg2

Assembly Language

See the TUCA-5.1 Assembly Language Reference Manual for information about programming for this architecture.