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 Assembly Language Reference Manual

This document describes the assembly language syntax for TTU Computer Architecture, version 5.1 (TUCA-5.1).

General Description

- The TUCA-5.1 assembly language is written in plain text, one instruction per line.
- The instruction operation is followed by various operands, separated by spaces.
- The order of the operands is the same as shown in the TUCA-5 Computer Architecture Specification.
- Registers are specified as “r0” to “r15”.
- Memory addresses are specified in hexadecimal, with the “0x” prefix.

Examples:

ld 0x01 r1	Loads from memory location 0x01 into register r1
add r0 r1 r2	Adds the values from registers r0 and r1 into r2.

Comments and Whitespace

- Any line beginning with the “#” character will be considered a comment line and ignored.
- Empty lines will be ignored.

Pre-processor Macro definitions

- Macro definitions are supported for replacing any part of an instruction with a more meaningful name
- This can be used to replace register names, memory addresses, etc.
- The format to define macros is: “def name value”
    - For example: “def counter r5”

Variables

- Named variables are supported in the form of macro definitions that can be used instead of a fixed memory address
- Variables need to be defined using the syntax “var name addr” at the top of the program

Examples:

def src 0x01
def dst 0x02
	Defines to variables "src" and "dst" to point to memory addresses 0x01 and 0x02 respectively

ld src r1	Loads from memory location 0x01 into register r1
add r0 r1 r2	Adds the values from registers r0 and r1 into r2.
st r2 dst	Stores the value in register r2 into memory address 0x02

Address labels and Jumps

Symbolic address labels can be added into a line by using the syntax: "label:" starting at the beginning of a line. Nothing else should follow on the same line.

Example:

loop1:	Defines the label "loop1"
add r0 r1 r0	Adds the values from registers r0 and r1 into r0.
skipif r2	Skip the next instruction if the value of r0 is non-zero.
jmp loop1	Jump to the instruction starting immediately following the "loop1" label (the add).

Dynamic Jumps

The jmpr instruction allows making jumps to addresses computed by the program. These are generally paired with the use of the loadpc instruction, to read the program counter.

Example:

Instruction Address	Instruction	Description
0x000	loadpc r5 r6	Store the current instruction address into r5 (hi) & r6 (lo)
0x002	jmp myfunc	Jump to label "myfunc"
0x004	halt	
0x006	myfunc:	
0x008	ldi 0x04 r4	Store the number 4 into r4
0x00A	ldi 0x01 r1	Store the number 1 into r1

0x00C	add r6 r4 r7	Do r5+4 into r2. This is the low part of the address + 4 bytes.
0x00E	gt r6 r7 r8	Check for overflow in the addition
0x010	if r8	If overflow
0x012	add r5 r1 r5	Add 1 to high part
0x014	jmpr r5 r7	Jump to the instruction at address stored in r5 (hi) & r7 (lo)