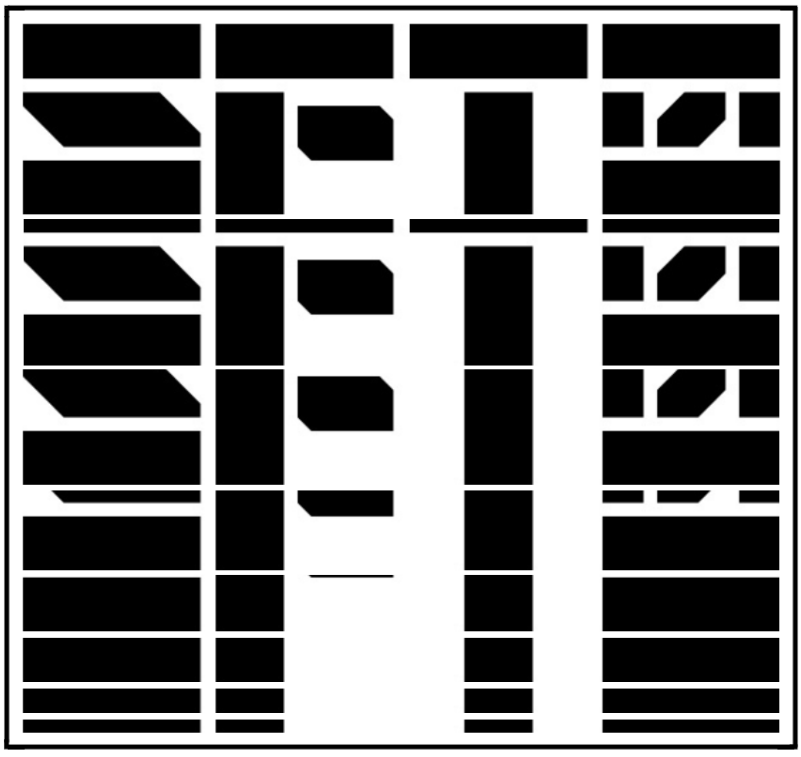
Archer 6969 Assembly Language Programming Manual



ARCHER INC. REV. A

ARC6969
MICROCOMPUTER FAMILY
PROGRAMMING MANUAL

Archer Computer Inc

August 27, 1969

Contents

Abstract

The information in this manual has been reviewed and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. The material in this manual is for informational purposes only and is subject to change without notice.

1 Introduction

This Manual describes the assembly language format, and how to write assembly language programs for the "ARC6969" microprocessor. Detailed information on the operation of specific assemblers is available in the Operator's Manual and Installation Guide for each specific assembler.

2 Computer Organization

- 32 8-bit registers
- 64KB of RAM
- 64x32 6-bit pixels, 60Hz screen
- Up to 7 external devices

2.1 Working Registers

The ARC6969 provides the programmer with 32 8-bit registers. These 32 working registers are numbered and referenced via the integers 0, 1, 2; by convention, these registers may also be accessed via the letters r0, r1, r2, ..., r31 respectively.

2.2 Program Counter

The Program Counter is a 16-bit special-purpose register used to sequence the execution of the program. It provides the address of the next instruction to be fetched from the RAM for execution. When a byte is fetched, then the PC is automatically incremented by one to point to the next memory location.

2.3 Flag Register

The Flag Register is the status register that contains the current state of the CPU. In ARC6969 architecture the register is 8 bits wide and represented as following:

7	6	5	4	3	2	1	0
0	0	0	0	OF	SF	CF	ZF

- ZF (Zero Flag) - This bit is set if result is zero.
- CF (Carry Flag) - This bit is set on high-order bit carry or borrow.
- SF (Sign Flag) - This bit is set equal to high-order bit of result (0 if positive 1 if negative).
- OF (Overflow Flag) - Set if result is too large a positive number or too small a negative number (excluding sign bit) to fit in destination operand.

2.4 Memory

The programmer visualizes memory as a sequence of bytes, each of which may store 8 bits (represented by two hexadecimal digits). The bits stored in a memory may represent the encoded form of an instruction or may be data. The address space is being between 0x0000 and 0xFFFF included.

3 Instruction Encoding

Every instruction is byte aligned in "Big Endian" format.
Terminology:
Rx/Ry/Rz - Any general purpose register (0, 1,.., 31).
Imm3 - Any 3bit value.
Imm8 - Any 8bit value.
Imm16 - Any 16bit value.

3.1 Arithmetic

Encoding:

Opcode	Unused	Rx	Ry	Imm8/Rz
5 bits	1 bit	5 bits	5 bits	8 bits

Opcode	Assembly	Description
0	add Rx, Ry, Rz	Rx = Ry + Rz
1	addi Rx, Ry, Imm8	Rx = Ry + Imm8
2	sub Rx, Ry, Rz	Rx = Ry - Rz
3	subi Rx, Ry, Imm8	Rx = Ry - Imm8
6	or Rx, Ry, Rz	Rx = Ry | Rz
7	ori Rx, Ry, Imm8	Rx = Ry | Imm8
8	xor Rx, Ry, Rz	Rx = Ry ^ Rz
9	xori Rx, Ry, Imm8	Rx = Ry ^ Imm8
10	and Rx, Ry, Rz	Rx = Ry & Rz
11	andi Rx, Ry, Imm8	Rx = Ry & Imm8
12	shl Rx, Ry, Rz	Rx = Ry <<Rz
13	shr Rx, Ry, Rz	Rx = Ry >>Rz

3.2 Comparison

Encoding:

Opcode	Unused	Rx	Unused	Imm8/Ry
5 bits	3 bits	5 bits	3 bits	8 bits

Opcode	Assembly	Description
4	cmp Rx, Ry	Compares Rx to Ry and sets the flags accordingly
5	cmpi Rx, Imm8	Compares Rx to Imm8 and sets the flags accordingly

Note: The flags can only be effected by CMP/CMPI instructions.
Flags affected:

Flag	Effect
ZF	1 if Rx equals Imm8/Ry
CF	1 if unsigned Imm8/Ry >unsigned Rx
SF	1 if MSB of Rx - Imm8/Ry <0
OF	1 if signed result of an operation is out of range

3.3 Control flow

Encoding:

Opcode	Unused	Imm16
5 bits	3 bits	16 bits

Opcode	Assembly	Description
24	call Imm16	Save next instruction address in R31 and jump to Imm16
25	ret	Jump to R31 register
16	jmp Imm16	Jump to Imm16
17	je Imm16	If ZF = 1 then jump to Imm16
18	jne Imm16	If ZF = 0 then jump to Imm16
19	jb Imm16	If CF = 1 then jump to Imm16
20	jl Imm16	If SF != OF then jump to Imm16
26	jg Imm16	If ZF = 0 and SF = OF then jump to Imm16
27	ja Imm16	If CF = 0 and ZF = 0 then jump to Imm16

3.4 Memory operations

In ARC6969 CPU all memory accesses are indirect and require 2 arguments to build 16-bit address.
Encoding:

Opcode	Unused	Rx	Ry	Unused	Rz
5 bits	1 bit	5 bits	5 bits	3 bits	5 bits

Opcode	Assembly	Description
14	rd Rx, Ry, Rz	Rx = [RyRz]
15	wr Rx, Ry, Rz	[RyRz] = Rx

Note: Ry - 8 most significant bits of memory address, while Rz - 8 least significant bits of memory address. RyRz builds 16 bit memory address.

3.5 IO device communication

The Device I/O interface allows communication with up to 7 external devices, selected by immediate Imm3. The Rx register is written to/read from depending on device. Accessing a device that does not exist halts the CPU.
Encoding:

Opcode	Unused	Rx	Imm3
5 bits	3 bit	5 bits	3 bits

Note: IO_GPU_UPDATE ignores all arguments. The Imm3 is an index in IoDevice table and encoded as following:

Index	IoDevice	Description
1	IO_GPU_SET_X	Set X position on GPU screen buffer
2	IO_GPU_SET_Y	Set Y position on GPU screen buffer
3	IO_GPU_DRAW	Set pixel on GPU screen buffer
4	IO_GPU_UPDATE	Refresh screen and draw buffer
5	IO_SERIAL_LENGTH	Return the number of bytes currently in serial buffer
6	IO_SERIAL_READ	Read 1 byte from serial buffer
7	IO_SERIAL_WRITE	Write 1 byte to serial out

Opcode	Assembly	Description
21	io Rx, Imm3	Read or Write to the IO device

The IO_GPU_DRAW Expects 6 bit color bitmap encoded as following:

5	4	3	2	1	0
Red	Red	Green	Green	Blue	Blue

4 Boot Process

The "ARC6969" loads its rom at memory address 0x0000, then the PC is set to 0 and execution begins. The execution starts with all registers being 0.