Grason’s CPU Reference Guide

v. 0.01

September 6, 2024

# Table of contents

[Table of contents 2](#_Toc176509277)

[Hardware Architecture 3](#_Toc176509278)

[System Diagram 3](#_Toc176509279)

[Instruction Pointer 3](#_Toc176509280)

[Temporary Register 3](#_Toc176509281)

[Address Buffer/Memory 4](#_Toc176509282)

[A and B Registers 4](#_Toc176509283)

[Arithmetic Logic Unit (ALU) 4](#_Toc176509284)

[Instruction Register 5](#_Toc176509285)

[Instruction Set 5](#_Toc176509286)

[SWP 5](#_Toc176509287)

[MOV A <immed> 5](#_Toc176509288)

[MOV A [MEM] 5](#_Toc176509289)

[SHL 5](#_Toc176509290)

[SHR 6](#_Toc176509291)

[INC A 6](#_Toc176509292)

[DEC A 6](#_Toc176509293)

[INC B 6](#_Toc176509294)

[DEC B 6](#_Toc176509295)

[AND B 7](#_Toc176509296)

[AND <immed> 7](#_Toc176509297)

[OR B 7](#_Toc176509298)

[OR <immed> 7](#_Toc176509299)

[XOR B 7](#_Toc176509300)

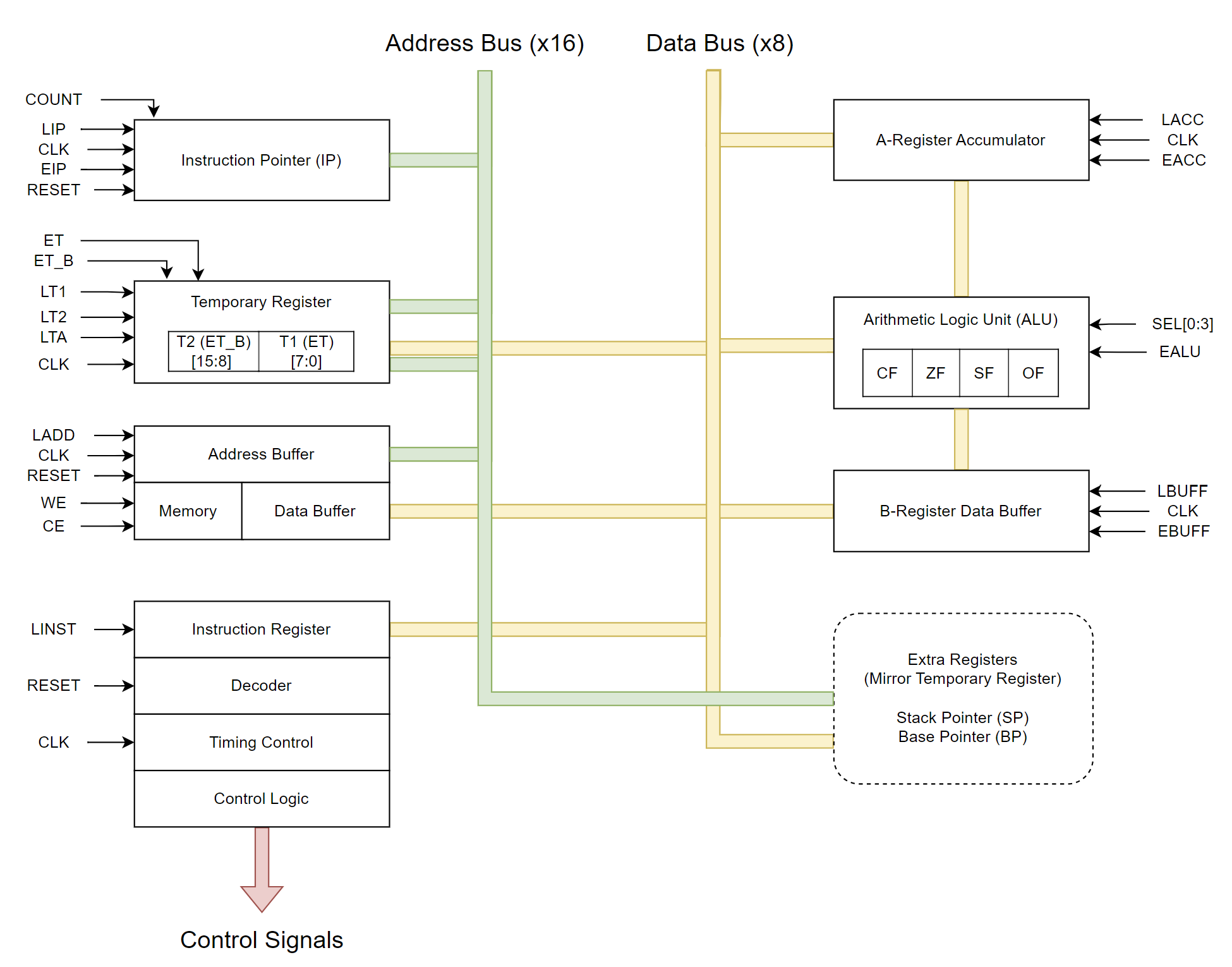
[XOR <immed> 8](#_Toc176509301)

[NOT 8](#_Toc176509302)

# 

# Hardware Architecture

## System Diagram



## Instruction Pointer

Description: Holds the address of the instruction currently loaded.

LIP: Load address bus to register.

EIP: Output register to address bus.

CLK: Increment IP by 1.

## Temporary Register

Description: Holds temporary data to be used elsewhere in the current operation

LT1: Load data bus to lower 8 bits [7:0].

LT2: Load data bus to upper 8 bits [15:8].

LTA: Load address bus to all 16 bits [15:0].

ET: Output register to address bus *and* output lower 8 bits [7:0] to data bus.

ET\_B: Output register to address bus *and* output upper 8 bits [15:8] to data bus.

## Address Buffer/Memory

Description: RAM and ROM memory for CPU and associated buffers.

LADD: Load address bus to address buffer.

RESET: Reset address buffer to 0x00.

WE: Write data bus to memory address in the address buffer.

CE: Output memory at address in the address buffer to data bus.

## A and B Registers

Description: Registers that feed the ALU. A receives the output of the ALU.

LACC/LBUFF: Load data bus to register.

EACC/EBUFF: Write from register to data bus.

## Arithmetic Logic Unit (ALU)

Description: Performs arithmetic and logic operations on data in A and B registers.

SEL[0:3]: Select which arithmetic/logic operation will be performed, based on following table:

|  |  |  |
| --- | --- | --- |
| **SEL[0:3]** | **Operation** | **Syntax** |
| 0x0 | Add | A + B |
| 0x1 | Subtract | A – B |
| 0x2 | Bit Shift Left | A << 1 |
| 0x3 | Bit Shift Right | A >> 1 |
| 0x4 | Increment A | A++ |
| 0x5 | Decrement A | A-- |
| 0x6 | Increment B | B++ |
| 0x7 | Decrement B | B-- |
| 0x8 | AND | A & B |
| 0x9 | OR | A | B |
| 0xA | XOR | A ^ B |
| 0xB | NOT A | ~A |
| 0xC | NOT B | ~B |
| 0xD | RESERVED |  |
| 0xE | RESERVED |  |
| 0xF | RESERVED |  |

EALU: Write EALU register to data bus.

CF: Carry Flag (CF) is set if result is incorrect for unsigned arithmetic.

OF: Overflow Flag (OF) is set if result is incorrect for signed arithmetic.

SF: Sign Flag (SF) is set if result is less than zero.

ZF: Zero Flag (ZF) is set if result is equal to zero.

## Instruction Register

Description: Holds opcodes and operands for current instruction. Performs all logic control flow for current instruction. This is where most of the magic happens.

LINST: Load data bus to register.

RESET: Return all outputs and state variables to initial values.

# Instruction Set

Each opcode/operand fetch takes 3 cycles and 1 memory byte

## SWP

Description: Swap A and B registers

Opcode: SWP && MOV (0x07)

Clock cycles: 7

Memory bytes: 1

## MOV A <immed>

Description: Move immediate operand value into A

Opcode: MOV && IMMEDA

Clock cycles: 8

Memory bytes: 2

## MOV A [MEM]

Description: Move value at memory into A

Opcode: MOV && MEMA

Clock cycles: 12

Memory bytes: 3

## SHL

Description: Bit shift left A register by value in B register

Opcode: LOG && SHL

Expected Operands: 0

ALU SEL: 0x2

Clock cycles: 7

Memory bytes: 1

## SHR

Description: Bit shift right A register by value in B register

Opcode: LOG && SHR

Expected Operands: 0

ALU SEL: 0x3

Clock cycles: 7

Memory bytes: 1

## INC A

Description: Increment A register

Opcode: LOG && INCA

Expected Operands: 0

ALU SEL: 0x4

Clock cycles: 7

Memory bytes: 1

## DEC A

Description: Decrement A register

Opcode: LOG && DECA

Expected Operands: 0

ALU SEL: 0x5

Clock cycles: 7

Memory bytes: 1

## INC B

Description: Increment B register

Opcode: LOG && INCB

Expected Operands: 0

ALU SEL: 0x6

Clock cycles: 7

Memory bytes: 1

## DEC B

Description: Increment A register

Opcode: LOG && DECB

Expected Operands: 0

ALU SEL: 0x7

Clock cycles: 7

Memory bytes: 1

## AND B

Description: Bitwise AND A and B register

Opcode: LOG && MEMA (0x21)

Expected Operands: 0

ALU SEL: 0x8

Clock cycles: 7

Memory bytes: 1

## AND <immed>

Description: Bitwise AND immediate value with A

Opcode: LOG && STORA (0x25)

Expected Operands: 0

ALU SEL: 0x8

Clock cycles:

Memory bytes: 2

## OR B

Description: Bitwise OR A and B register

Opcode: LOG && MEMB (0x22)

Expected Operands: 0

ALU SEL: 0x9

Clock cycles: 7

Memory bytes: 1

## OR <immed>

Description: Bitwise OR immediate value with A

Opcode: LOG && STORB (0x26)

Expected Operands: 0

ALU SEL: 0x9

Clock cycles:

Memory bytes:

## XOR B

Description: Bitwise XOR A and B register

Opcode: LOG && IMMEDA (0x23)

Expected Operands: 0

ALU SEL: 0xA

Clock cycles: 7

Memory bytes: 1

## XOR <immed>

Description: Bitwise XOR immediate value with A

Opcode: LOG && SWP (0x27)

Expected Operands: 0

ALU SEL: 0xA

Clock cycles:

Memory bytes:

## NOT

Description: Bitwise NOT A register

Opcode: LOG && IMMEDB (0x24)

Expected Operands: 0

ALU SEL: 0xB

Clock cycles: 7

Memory bytes: 1

## CALL

Description: Save IP to SP, increment SP, set IP to given address

Opcode: JMP && STORA (0x85)

Syntax: CALL [MEM]

Clock cycles:

Memory bytes: 3