Developer's Guide – Full Edition

0. Table of contents

1. PI architecture overview

Core principles:

1. Simplicity – all CPU **registers are co-addressed as memory**, giving the programmer maximum wiggle room in manipulating the data. There are only 2 instructions and the ISA document is short, making the assembly easy to learn and understand.

2. Complexity on the programmer's side – you, as the programmer are the best equipped to know how data should be manipulated by your code. That's why there's **no complex multi-step instructions**. there is no micro-code – **PI assembly itself is essentially the micro-code**. This way you have maximum control over what exactly is being done – isn't this exactly what coding in assembly is about?

3. Speed – **Every instruction takes one clock cycle** and one clock cycle only. Architecture is also designed, to enable implementation of a pipeline.

General characteristics:

The architecture has 16 8-bit registers; Instruction Pointers, 3 registers for ALU operations and 12 general-purpose registers. The architecture supports up to 10-bit memory addresses.

There are 4 write-only flags, that dictate behaviour of the processor.

Arithmetical-Logical Unit (ALU):

This most crucial element of the CPU conducts arithmetical (addition and subtraction) and logical (NOR and NAND) operations.
In the 8-bit PI architecture CPU, the **ALU has two 8-bit inputs and one 8-bit output.**

Whenever ,,ALU input" and ,,ALU output" is mentioned in this document, it is referring to those values.
Carry bit from the last operation arithmetical operation is also always stored in the ALU.

2. Registers, memory and flags

All registers and memory cells are 8-bit wide.

Memory:

All CPU registers are memory-mapped. Meaning that they all can be accesed just like normal memory cells.

	Address:	Annotation:
Special Registers	0000**0000**	**0000** registry – Instruction Pointer
	0000**0001**	**0001** registry - Alpha
	0000**0010**	**0010** registry - Beta
	0000**0011**	**0011** registry - Accumulator
General Purpose Registers	0000**0100**	**0100** registry – Registry 4
	...	
	0000**1110**	**1110** registry – Registry 14
	0000**1111**	**1111** registry – Registry 15
Regular memory	00010000	
	…	
	11111110	
	11111111	

The last 4 bits of the memory address correspond to the registry number.

Instruction Pointer (IP) – shows instruction that is currently being executed. 2 is added automatically after each instruction to this registry. IP can only store even numbers.

Alpha and Beta – two registers wired directly to the ALU at all times (always enabled).

Accumulator – hard wired to ALU, after each instruction in which Alpha or Beta are changed the Accumulator is automatically updated with ALU output.

Memory containing instructions shall always be aligned by 2. First byte of the instruction is **to be stored under an even address**. Next byte is to be stored in the next memory cell (with an odd address). As shown below:

Address:	Content:
XXXXXXX0	Instruction 1
XXXXXXX1	
XXXXXXX0	Instruction 2
XXXXXXX1	

Because all registers serve as memory, there is nothing preventing you from storing instructions inside registers.

Flags:

Flag number:	Flag name:	Description:
00	HF – Halt Flag	When the flag is set to 1, all activity of the CPU is stopped. When the flag is set to 0, CPU works normally.
01	SF – Skip Flag	When the flag is set to 1, all instructions are skipped, except instructions that would change the value of this flag. When the flag is set to 0 - all instructions are executed normally.
10	AF0 – ALU Flag 0	*See below*
11	AF1 – ALU Flag 1	*See below*

Interpretation of ALU flag values:

AF1	AF0	ALU operation
0	0	Sum *(Alpha + Beta)*
0	1	Difference *(Alpha - Beta)*
1	0	NAND of Alpha and Beta
1	1	NOR of Alpha and Beta

3. Instructions

All instructions are 2-byte long, meaning they take 2 memory cells each.

MOV:

The most important and basic instruction of the PI assembly. It is used to move data from a registry to memory (or reverse). Note, that since all registers are co-addressed with memory – you can also move data between registers (**MOV 0 0001 00000011** will move data from the 0011 registry to the 0001 registry).

0	**R**	**REG**	**MEM-ex**	**MEM**
1 bit	1 bit	4 bits	2 bits	8 bits
Instruction code	Reverse Flag	Registry number	Extended memory address	Memory address

When R is 0, data is transferred from memory (MEM) to a registry (REG). When R is 1, the direction is reversed.

Aside from the standard 1-byte memory address, 2 additional bits are available (MEM-ex), if the machine has more than 256 bytes of memory.

FLAG:

This is an instruction, that serves to alter the control flow of the program, by changing flags (described on page 2).

1	FLAG	VALUE	CARRY	MEM	OP	NEG	ADR
1 bit	2 bits	1 bit	1 bit	1 bit	1 bit	1 bit	4 bits
Instruction code	Flag number	Value inserted into the flag	Indicating if to include carry from the last operation	Indicating if to include a OR of a memory cell	Indicating the operation to be performed on the inputs (0 – OR, 1 - AND)	If set to 1 – the final value inserted into the flag will be negated	Memory address, that will be included (of REG is set to 0, this field is ignored)

Flag indicated by the argument **FLAG** will be set.

Other arguments determine what will the input be.

VALUE is given directly to be put into the FLAG

CARRY and MEM indicate if **Carry from the ALU (from the last operation)** and l**ogical sum (OR) of all bits under a given memory address (ADR)** will be combined with the value given in argument VALUE.

OP indicates in what way all included values will be combined (AND / OR).

NEG gives the opportunity to negate the whole input (after including VALUE, CARRY and ACC, as well as conducting operation specified by OP).