The PenIS-Architecture

1. Introduction

The PenInstructionSet-Architecture is a proof of concept idea to create a fictional instruction set from the ground up. PenInstructionSet-Architecture‘s idea came when tinkering around with a 6502 Assembler and wondering what could be done to make a fun to write and flexible instruction set from the ground up. Due to the funny and to some extend controversial naming of the project you will of course need to realize, that this is nowhere near a big and organized project with hundreds of people working on it, but rather a small project to do in the creators spare time. The PenIS-A aims to not inherit many concepts of other popular instruction set architectures out there, namely x86, ARM or RISC-V. The project is not to be taken seriously and could be discontinued at any point due to lack of interest in it, so don’t build anything substantial on top of it or expect frequent and updates.

2. Conventions

This section of the PenInstructionSet-Architecture manual describes the different terminology and conventions used in the following chapters.[1]

2.1 Definitions

- PenIS referring to the PenInstructionSet.
- PenIS-A referring to the PenInstructionSet-Architecture.
- Register referring to a register inside of the CPU.
- UIP referring to the Underlying Initialization Phase of the CPU after startup.
- GPR referring to General-Purpose-Register.
- NVUIPM referring to the Non-Volatile UIP Memory.
- RMAPPED referring to the contents of one register directly affecting the contents of another register (e.g. setting bit 4 in the SEXC register will also set bit 4 in the SEX register).

2.2 Conventions

Individual bits of described registers can have the following types:

RW – The bit can be read/ written to.
RO – The bit is read-only.
WO – The bit is write only.
RES / UNU – The bit is currently unused or reserved.[2]

UNDEF – The contents of this bit are undefined.
USED – The bit is used by other parts of the architecture and thus should not be used at all.

Registers are described the following way:

Name	Width	Description

The individual bits of a register are described the following way:

last bit:first bit	Accessibility	Name of the bit[3]	Value after completion of the UIP

1 **Note:** This may very well be expanded upon in further releases of this manual and is thus likely to not cover/ forget to cover some important terminology/ conventions.
2 **Note:** Reserved bits and unused bits are practically the same. Both can be read or written to without any side effects, but it is advised to not make use of them in a non-testing matter.
3 **Note:** If there are only 3 fields this means that the bits have no name.

3 Register overview

3.1 General purpose registers

CUM: This register defines the Accumulator register used as a output of certain arithmetic instructions.
RE1 – RE15: The 15 64-bit GPRs.

3.2 Read-only system descriptor registers

SEX: This register defines the currently supported Standard Architectural Extensions.
SCI: This register stores the first memory address of user code that gets jumped to after the UIP.

3.3 Modifyable system descriptor registers

SEXC: This register defines the Standard Architectural Extensions Control used to deactivate certain SEXs in the SEX register.[4]

3.4 Additional registers

PORN: This register defines the Port Output Register for port N of the CPU(E.g. to write to Port 10, the register for it would be POR10).
IRN: This register defines the Input Register for port N of the CPU(E.g. to read from Port 10, the register for it would be IR10).

3.5 Register info

3.5.1 Accumulator

Name	Width	Description
Accumulator	64 bits	The accumulator is the first GPR in the list and is also the most used one when it comes to arithmetic instructions.

Bit layout:

64:0	RW	UNDEF

4 **Note:** Bits marked as RO in the SEX register mustn‘t be set by the programmer in the SEXC register.

3.5.2 General purpose registers (GPRs)

Name	Width	Description
REG1-REG15	64 bits	All general purpose registers are supposed to each behave the exact same.

Bit layout:

64:0	RW	UNDEF

3.5.3 Standard Architectural Extensions (SEX)

Name	Width	Description
SEX	64 bits	The SEX defines the standard architectural extensions currently in use by the CPU.

Bit layout:

64:4	RES	Reserved[5]	UNDEF
3	RO	MEME	UNDEF
2	RO	FLP_ENABLE	UNDEF
1	RO	FLOATING_POINT_PRESENT	UNDEF
0	RO	STD_PENIS	Always 1

Bit description:

MEME: Specifies if the CPU supports MEME extensions described later in the manual.
FLOATING_POINT_PRESENT: Specifies if the CPU supports floating point calculation.
FLP_ENABLE: Specifies if floating point calculations are enabled by default.
STD_PENIS: Specifies if the CPU supports the entirety of the PenIS and is thus always set to one by the UIP. If this bit may be set to 0 the programmer is advised to temporarily store the contents of the SEX in the NVUIPM and issue a reset by setting bit 5 or bit 6 in the SEXC.

3.5.4 Standard Architectural Extensions Control (SEXC)

Name	Width	Description
SEXC	64 bits	RMAPPED register mapped on top of the SEX register used to enable specific architectural extensions.

5 **Note:** Some of these bits are accessable through the SEXC register.

Bit layout:

64:7	RES	Reserved	UNU
6	WO	UIP_RESTART	Always 0
5	WO	HARD_RESTART	Always 0
4	WO	MAN_FLPE	UNDEF
3:0	RES	Reserved	UNDEF[6]

Bit description:

UIP_RESTART: Redo the UIP startup routine.
HARD_RESTART: Hard restart the CPU.
MAN_FLPE: Manually enable floating point extensions, if supported.

3.5.5 Port Output Register for port N (PORN)

3.5.6 Input Register for port N (IRN)

6 These bits are natively used by the SEX register but aren‘t guaranteed to hold the same information as the ones in the SEX register.

4. Instruction set reference

Instructions in the PenInstructionSet-Architecture need to consist of a minimum of 3 bytes, with the first byte being the opcode of the current instruction and the second and third one being the so called “operand bytes”, which specifies the behavior of the current instruction. The following bytes will be used as operands for the current opcode. Each instruction can have a maximum of 2 and a minimum of 0 operands.

4.1 Operand bytes

The operand bytes (also referable to as the “operand word”) is laid out in 4-bit sections in the following way:

15:12 (Accessing mode)	11:8 (Operand size)	7:4 (Operand 1)	3:0 (Operand 2)

Here is a table containing the different values of the 4 4-bit sections and their corresponding meanings. The reason that the Operand 1 and Operand 2 fields are not present in the table is because the value of the Operand 1/ Operand 2 corresponds to one of the 16 GPRs. If the accessing mode is set to Mem-Mem, Mem-Reg or Reg-Mem, the Mem operands are ignored in their respective positions in the Operand 1 or Operand 2 fields and are placed in the bytes following the operand bytes.

Value/ Field	Accessing mode	Operand size
0000	Reg	64-bit
0001	Reg-Reg	32-bit
0010	Reg-Mem	16-bit
0011	Mem-Reg	8-bit
0100	Mem-Mem	24-bit (only supported when activating the MEME extensions)
0101	Immediate	32-bit
0111	None	64-bit

4.2 Transfer instructions

Name	Instruction syntax	Opcode	Supported operand sizes	Supported accessing modes
STICK	STICK <oper1>, <oper2>	0x1	64-bit, 32-bit, 16-bit, 8-bit, 24-bit	Reg-Reg, Reg-Mem, Mem-Reg, Mem-Mem

4.3 Individual instruction descriptions

STICK: Sticks the content of the second operand into the first operand. This instruction has to be supported by every revision of the PenInstructionSet.

5. Additional notes

:^)