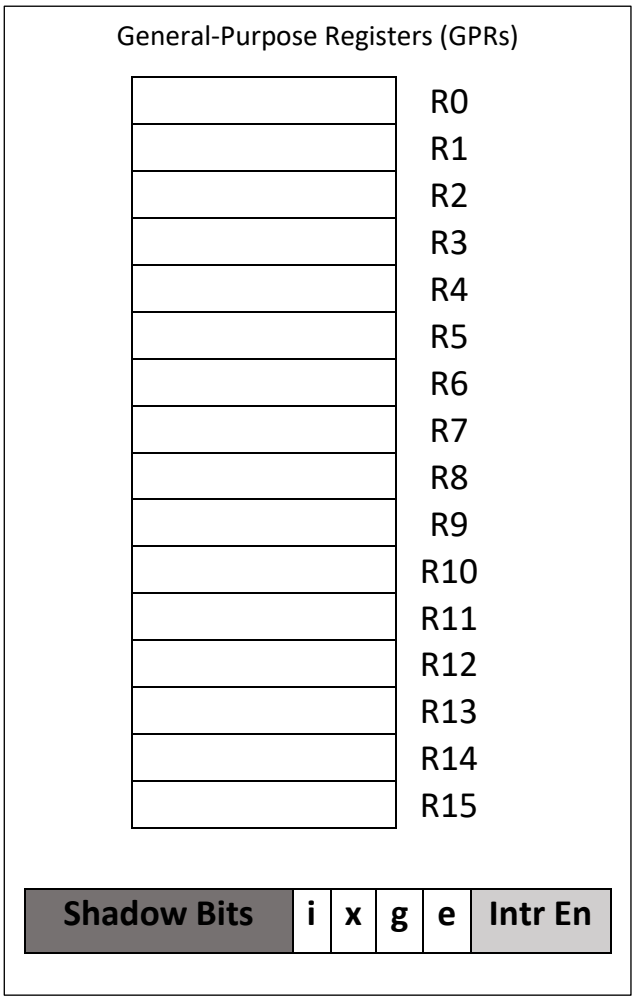
SAYAC Datasheet

Document Version 0.9

Contributors to all versions of the spec:
Professor Zain Navabi, Katayoon Basharkhah and Hanieh Totonchi

SAYAC is a Simple architecture Yet Ample Circuitry that supports 16-bit operations. Despite the limited number of registers in this processor, it can support almost all necessary instructions for performing the vast body of existing applications. SAYAC includes features which make it particularly suitable for the purpose of tutoring and research.

Registers



Instruction Summary

[15:12]	[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3:0]	Instruction	Notation
0000	Reserved										
0001	Reserved										
0010	00		0		rs1				rd	LDR	Rf(rd) <= MEM(Rf(rs1))
			1								Rf(rd) <= IO(Rf(rs1))
	01		0		rs1					STR	MEM(Rf(rd)) <= Rf(rs1)
			1								IO(Rf(rd)) <= Rf(rs1)
	10		s		rs1					JMR	PC <= PC + Rf(rs1) Rf(rd) <= PC + 1, IF s=1
	11		imm						rd	JMI	PC<= PC + imm rd <= PC + 1
0011	rs2				rs1				rd	ANR	Rf(rd) <= Rf(rs1) AND Rf(rs2)
0100	imm								rd	ANI	rd <= rd AND USE(imm)
0101	imm								rd	MSI	rd <= SE(imm)
0110	imm								rd	MHL	rd [15:8] <= imm
0111	rs2				rs1				rd	SIR	Rf(rd) <= Rf(rs1) LS± Rf(rs2)
1000	rs2				rs1				rd	SAR	Rf(rd) <= Rf(rs1) AS± Rf(rs2)
1001	rs2				rs1				rd	ADR	Rf(rd) <= Rf(rs1) + Rf(rs2)
1010	rs2				rs1				rd	SUR	Rf(rd) <= Rf(rs1) – Rf(rs2)
1011	imm								rd	ADI	rd <= rd + SE(imm)
1100	imm								rd	SUI	rd <= rd - SE(imm)
1101	rs2				rs1				rd	MUL	Rf(rd) <= Rf(rs1) × Rf(rs2) 'LSB Rf(rd+1) <= Rf(rs1) × Rf(rs2) 'MSB
1110	rs2				rs1				rd	DIV	Rf(rd) <= Rf(rs1) ÷ Rf(rs2) 'Quo Rf(rd+1) <= Rf(rs1) ÷ Rf(rs2) 'Rem
1111	00		0		rs1				rs1	CMR	flags <= Cmp(Rf(rs1), Rf(rd))
			1	imm					rs1	CMI	flags <= compare(rs1,SE(imm))
	01		0	flag interpretation bits					rd	BRC	PC <= rd if flag is active
			1	flag interpretation bits					rd	BRR	PC <= PC + rd if flag is active
	10		0	shim					rd	SHI	rd <= rd LS± shim
			1	shim					rd		rd <= rd AS± shim
	11		0	0	rs1				rd	NTR	Rf(rd) <= 1sComp(Rf(rs1))
				1							Rf(rd) <= 2sComp(Rf(rs1))
			1	0						NTD	rd <= 1's complement(rd)
				1							rd <= 2's complement(rd)

Instruction Set Reference

The following pages list all SAYAC instruction mnemonics:

Word Formats:

There are two types of SAYAC instruction word format: I-type, R-type.

I-Type Instruction:

I-type instruction word format contains an immediate value embedded within the instruction word.

R-Type Instruction:

R-type instruction word format contains registers for input and results.

LDR/LDRio

Opcode				Opcode-Extended		Memory/IO		r_{s1}				r_d			
0	0	1	0	0	0	0/1									

LdR / LdRio	**Load Registered**
Instruction	load from memory or I/O peripheral
Operation	$.r_d. \leftarrow [.r_{s1}.]$ or $\{.rs_1.\}$
Assembler Syntax	LdR r_d r_{s1}
Example	LdR r3 r2
Description	Loads register r_d with the desired memory byte at the address specified with r_{s1}. If Memory/IO bit is one, then it bypasses the memory transfer
Instruction Type	R
Instruction Fields	r_{s1}= Index of source register r_d= Index of destination register Memory/IO= selection bit for memory or I/O peripheral

STR/STRio

Opcode				Opcode-Extended		Memory/IO		r_{s1}				r_d			
0	0	1	0	0	1	0/1									

STR / STRio	**Store Registered**
Instruction	Store to memory or I/O peripheral
Operation	$[.r_{s1}.]$ or $\{.rs_1.\} \leftarrow r_{s1}$
Assembler Syntax	STR r_d r_{s1}
Example	STR r3 r2
Description	Stores register r_{s1} to the memory location specified with r_d. If Memory/IO bit is one, then it bypasses the memory transfer
Instruction Type	R
Instruction Fields	r_{s1}= Index of source register r_d= Index of destination register Memory/IO= selection bit for memory or I/O peripheral

JMR

Opcode				Opcode-Extended		Save PC		r_{s1}				r_d			
0	0	1	0	1	0	s									

JMR	**Jump Registered (Unconditional)**
Instruction	Jump to address
Operation	PC ← PC + .r_{s1}. .r_d. ←PC+1 if s=1
Assembler Syntax	JMR r_d r_{s1}
Example	JMR r3 r2
Description	Transfers execution to the address contained in register r_{s1} relative to the current instruction pointer. Saves the address of the next instruction in register r_d if the Save PC bit (s) equals to 1. This option is used when returning from interrupts and exceptions.
Instruction Type	R
Instruction Fields	r_{s1}= Index of source register r_d= Index of destination register s= Option for saving PC contents

JMI

Opcode				Opcode-Extended		Imm [5:0]						r_d			
0	0	1	0	1	1										

JMI	**Jump Immediate (Unconditional)**
Instruction	Jump to immediate address
Operation	PC ← PC + SE"Imm" .r_d. ←PC+1
Assembler Syntax	JMI r_d Imm
Example	JMI r3 32
Description	Transfers execution to the address contained in immediate value relative to the current instruction pointer and saves the address of the next instruction in register r_d.
Instruction Type	I
Instruction Fields	r_d= Index of destination register Imm = 6-bit signed immediate value

ANR

Opcode				r_{s1}				r_{s2}				r_d			
0	0	1	1												

ANR	**AND Registered**
Instruction	Logical AND operation
Operation	$.r_d. \leftarrow .r_{s1}.$ AND $.r_{s2}.$
Assembler Syntax	ANR r_d r_{s1} r_{s2}
Example	ANR r5 r3 r2
Description	Calculates the bitwise logical AND of r_{s1} and rs2 and stores the result in r_d .
Instruction Type	R
Instruction Fields	r_{s1}= Index of source1 register r_{s2}= Index of source2 register r_d= Index of destination register

ANI

Opcode				Imm[7:0]								r_d			
0	1	0	0												

ANI	**AND Immediate**
Instruction	Logical AND operation
Operation	$.r_d. \leftarrow .r_d.$ USE"Imm"
Assembler Syntax	ANI r_d Imm
Example	ANI r5 250
Description	Calculates the bitwise logical AND of r_d and 16-bit concatenated Immediate value, ((0x00) & Imm), and stores the result in r_d.
Instruction Type	R
Instruction Fields	r_d= Index of destination register Imm = 8-bit unsigned immediate value

MSI

Opcode				Imm[7:0]								r_d			
0	1	0	1												

MSI	**Move Signed Immediate**
Instruction	Move low sign extended immediate to register
Operation	$.r_d. \leftarrow$ SE"Imm"
Assembler Syntax	MSI r_d Imm
Example	MSI r5 100

Description	Writes the immediate value, Imm, into the low halfword of r_d, and sign extends the higher halfword of r_d. Writes the singed extention of the immediate value, Imm, into r_d.
Instruction Type	I
Instruction Fields	r_d= Index of destination register Imm = 8-bit unsigned immediate value

MHI

Opcode				Imm[7:0]								r_d			
0	1	1	0												

MHI	**Move High Immediate**
Instruction	Move high immediate to register
Operation	.r_d.'MSB ← Imm
Assembler Syntax	MHI r_d Imm
Example	MHI r5 100
Description	Writes the immediate value, Imm, into the high halfword of r_d keeping its low halfword.
Instruction Type	I
Instruction Fields	r_d= Index of destination register Imm = 8-bit unsigned immediate value

SLR

Opcode				r_{s1}				r_{s2}				r_d			
0	1	1	1												

SLR	**Shift Logical Registered**
Instruction	Logical Left/Right shift
Operation	.r_d. ← .r_{s1}. << (± .r_{s2}.[4:0])
Assembler Syntax	SLR r_d r_{s1} r_{s2}
Example	SLR r_5 r_2 r_3
Description	Shifts r_{s1} by the number of bits specified in r_{s2}[4:0][3:0] and then stores the result in r_d. Based on the sign of r_{s2} [4:0] left (-) or right (+) will be performed.
Instruction Type	R
Instruction Fields	r_d= Index of destination register r_{s1}= Index of source1 register r_{s2}= Index of source2 register

SAR

Opcode				r_{s1}				r_{s2}				r_d			
1	0	0	0												

SAR	**Shift Arithmetic Registered**
Instruction	Arithmetic Left/Right shift
Operation	$.r_d. \leftarrow .r_{s1}. <<< (\pm .r_{s2}.[4:0])$
Assembler Syntax	SAR r_d r_{s1} r_{s2}
Example	SAR r_5 r_2 r_3
Description	Shifts r_{s1} by the number of bits specified in $r_{s2}[4:0][3:0]$ and then stores the result in r_d. Based on the sign of r_{s2} [4:0] left (-) or right (+) will be performed.
Instruction Type	R
Instruction Fields	r_d= Index of destination register r_{s1}= Index of source1 register r_{s2}= Index of source2 register

ADD

Opcode				r_{s1}				r_{s2}				r_d			
1	0	0	1												

ADD	**Add Registered**
Instruction	Adding two registers
Operation	$.r_d. \leftarrow .r_{s1}. + .r_{s2}.$
Assembler Syntax	ADD r_d r_{s1} r_{s2}
Example	ADD r_5 r_2 r_3
Description	Calculates the sum of r_{s1} and r_{s2}. Stores the result in r_d. Used for unsigned addition.
Instruction Type	R
Instruction Fields	r_d= Index of destination register r_{s1}= Index of source1 register r_{s2}= Index of source2 register

SUB

Opcode				r_{s1}				r_{s2}				r_d			
1	0	1	0												

SUB	**SUB Registered**
Instruction	Subtracting two registers
Operation	$.r_d. \leftarrow .r_{s1}. - .r_{s2}.$
Assembler Syntax	SUB r_d r_{s1} r_{s2}
Example	SUB r_5 r_2 r_3

Description	Subtract r_{s2} from r_{s1} and store the result in r_d. Used for unsigned subtraction.
Instruction Type	R
Instruction Fields	r_d= Index of destination register r_{s1}= Index of source1 register r_{s2}= Index of source2 register

ADI

Opcode				Imm[7:0]								r_d			
1	0	1	1												

ADI	**ADD Immediate**
Instruction	Adding Immediate to register
Operation	$.r_d. \leftarrow .r_d. + SE"Imm"$
Assembler Syntax	ADI r_d Imm
Example	ADI r_5 150
Description	Sign-extends the 8-bit immediate value and adds it to the value of r_d. Stores the sum in r_d.
Instruction Type	I
Instruction Fields	r_d= Index of destination register Imm = 8-bit unsigned immediate value

SUI

Opcode				Imm[7:0]								r_d			
1	1	0	0												

SUI	**SUB Immediate**
Instruction	Subtracting Immediate from register
Operation	$.r_d. \leftarrow .r_d. - SE"Imm"$
Assembler Syntax	SUI r_d Imm
Example	SUI r_5 150
Description	Sign-extends the 8-bit immediate value and subtracts it from the value of r_d. Stores the result in r_d.
Instruction Type	I
Instruction Fields	r_d= Index of destination register Imm = 8-bit unsigned immediate value

MUL

Opcode				r_{s1}				r_{s2}				r_d			
1	1	0	1												

MUL	**Multiply Registered**
Instruction	Multiplying two registers
Operation	$.r_d. \leftarrow .r_{s1}.* .r_{s2}.$ 'LSB $.r_d+1. \leftarrow .r_{s1}.* .r_{s2}.$ 'MSB
Assembler Syntax	MUL r_d r_{s1} r_{s2}
Example	MUL r_5 r_3 r_2
Description	Multiplies r_{s1} times r_{s2} and stores the 16 low-order bits of the product to r_d and the high-order bits of the product to r_d+1.
Instruction Type	R
Instruction Fields	r_d= Index of destination register r_{s1}= Index of source1 register r_{s2}= Index of source2 register

DIV

Opcode				r_{s1}				r_{s2}				r_d			
1	1	1	0												

DIV	**Divide Registered**
Instruction	Dividing two registers
Operation	$.r_d. \leftarrow .r_{s1}. \div .r_{s2}.$ 'Quo $.r_{d+}1. \leftarrow .r_{s1}. \div .r_{s2}.$ 'Rem
Assembler Syntax	DIV r_d r_{s1} r_{s2}
Example	DIV r_5 r_3 r_2
Description	Divides r_{s1} by r_{s2} and then stores the integer portion of the resulting quotient to r_d and the resulting remainder to r_d+1.
Instruction Type	R
Instruction Fields	r_d= Index of destination register r_{s1}= Index of source1 register r_{s2}= Index of source2 register

CMR

Opcode				Opcode-Extended				r_{s2}				r_{s1}			
1	1	1	1	0	0	0									

CMR	**Compare Registered**
Instruction	Comparing two registers

Operation	If ($.r_{s2}. > .r_{s1}.$) then G←1
	If ($.r_{s2}. < .r_{s1}.$) then L←1
	If ($.r_{s2}. = .r_{s1}.$) then E←1

Assembler Syntax	CMR r_{s1} r_{s2}
Example	CMR r_3 r_2
Description	Compares r_{s1} and r_{s2} and stores the comparison result in the corresponding flags
Instruction Type	R
Instruction Fields	r_{s1}= Index of source1 register r_{s2}= Index of source2 register

ANI

Opcode				Opcode-Extended			Imm[4:0]					r_{s1}			
1	1	1	1	0	0	1									

CMI	**Compare Immediate**
Instruction	Comparing two registers

Operation	If (SE"Imm" > r_{s1}) then G←1
	If (SE"Imm" < r_{s1}) then L←1
	If (SE"Imm" = r_{s1}) then E←1

Assembler Syntax	CMI Imm r_{s1}
Example	CMI 30 r_2
Description	Sign-extends the 5-bit immediate value to 16 bits and compares it to the value of rs_1 and stores the comparison result in the corresponding flags.
Instruction Type	I
Instruction Fields	r_{s1}= Index of source1 register Imm = 5-bit unsigned immediate value

BRC

Opcode				Opcode-Extended			FIB					r_d			
1	1	1	1	0	1	0									

BRC	**Branch Conditional**
Instruction	Branch Registered with Condition
Operation	If (FIB) then PC ← .r_d.
Assembler Syntax	BRC FIB r_d
Example	BRC 0x01 r_2
Description	If Flag Interpretation Bits (FIB) are true, then transfers program control to the instruction at the address specified by register r_d
Instruction Type	R
Instruction Fields	r_d= Index of destination register FIB = 5-bit compare flag interpretation bits

BRR

Opcode				Opcode-Extended			FIB					r_d			
1	1	1	1	0	1	1									

BRR	**Branch Conditional Relative**
Instruction	Branch Registered Relative with Condition
Operation	If (FIB) then PC ← PC + .r_d.
Assembler Syntax	BRR FIB r_d
Example	BRR 0x01 r_2
Description	If Flag Interpretation Bits (FIB) are true, then transfers program control to the instruction to the address contained in register r_d relative to the current instruction pointer.
Instruction Type	R
Instruction Fields	r_d= Index of destination register FIB = 5-bit compare flag interpretation bits

SHI

Opcode				Opcode-Extended		LA	shim[4:0]					r_d				
1	1	1	1	1	0	0/1										

SHI	**Shift Arithmetic/Logical Immediate**
Instruction	Arithmetic Logical shift with immediate
Operation	$.r_d. \leftarrow .r_d. <<(\pm$ shim) if (LA=0) $.r_d. \leftarrow .r_d. <<<(\pm$ shim) if (LA=1)
Assembler Syntax	SAR r_d r_{s1} r_{s2}
Example	SAR r_5 r_2 r_3
Description	Shifts r_d by the number of bits specified in shim and then stores the result in r_d. Based on the sign of shim left (-) or right (+) will be performed. If the value of LA equals to zero and one, logical and arithmetic shift is done respectively.
Instruction Type	I
Instruction Fields	r_d= Index of destination register shim=5-bit immediate value for shift

NTR

Opcode				Opcode-Extended			1/2C	r_{s1}				r_d			
1	1	1	1	1	1	0	0/1								

NTR	**Not Registered**
Instruction	Logical NOT
Operation	$.r_d. \leftarrow$ 1's complement $(.r_{s1}.)$ if (1/2C=0) $.r_d. \leftarrow$ 2's complement $(.r_{s1}.)$ if (1/2C=1)
Assembler Syntax	NTR r_d r_{s1}
Example	NTR r_5 r_2
Description	Based on the value of 1/2C being 0 and 1, calculates 1's and 2's complement of register r_{s1} respectively and stores the value in register r_d
Instruction Type	R
Instruction Fields	r_d= Index of destination register r_{s1} = Index of source register 1/2C=Selection between 1's and 2's complement

NTD

Opcode				Opcode-Extended			1/2C					r_d			
1	1	1	1	1	1	1	0/1								

NTD	**Not Registered**
Instruction	Logical NOT
Operation	$.r_d. \leftarrow$ 1's complement ($.r_d.$) if (1/2C=0) $.r_d. \leftarrow$ 2's complement ($.r_d.$) if (1/2C=1)
Assembler Syntax	NTD r_d
Example	NTD r_5
Description	Based on the value of 1/2C being 0 and 1, calculates 1's and 2's complement of register r_d respectively and stores the value in register r_d
Instruction Type	R
Instruction Fields	r_d= Index of destination register 1/2C=Selection between 1's and 2's complement