Lecture 17

PicoBlaze Overview

Required reading

- P. Chu, *FPGA Prototyping by VHDL Examples*

  Chapter 14, PicoBlaze Overview

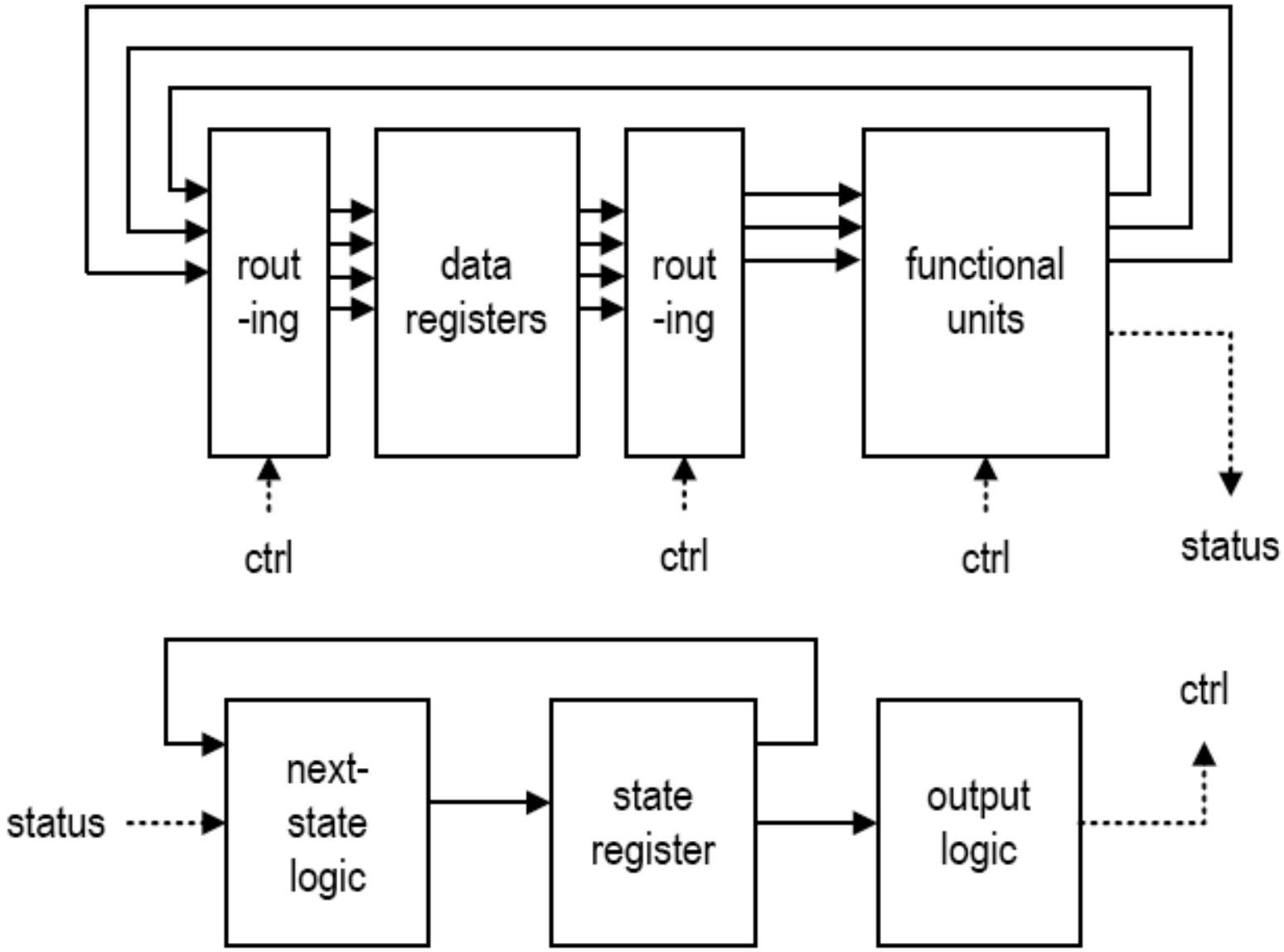
Recommended reading

PicoBlaze 8-bit Embedded Microcontroller User Guide

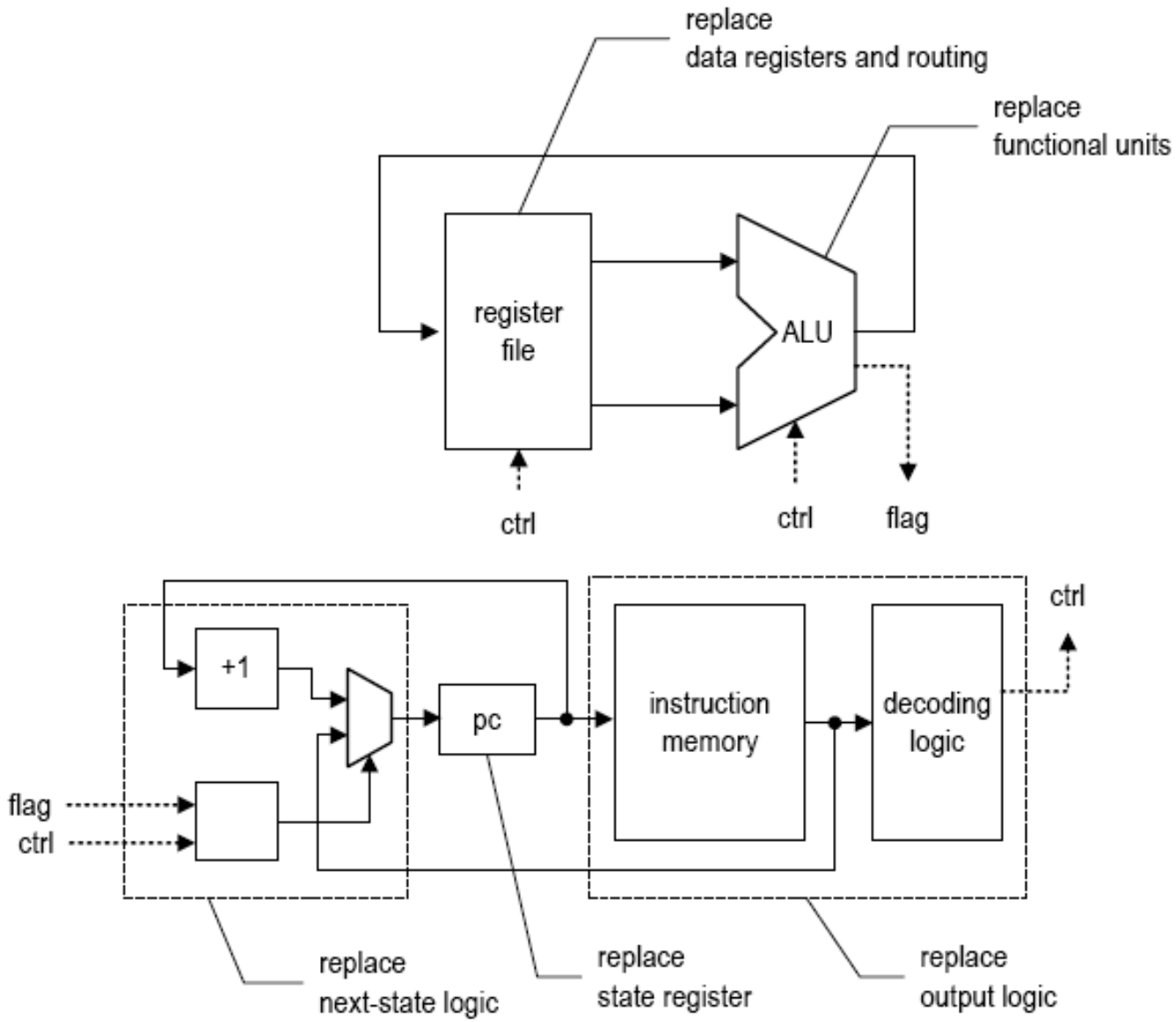
for Spartan-3, Virtex-II, and Virtex-II Pro FPGAs

(search for it using Google or Xilinx website documentation search)

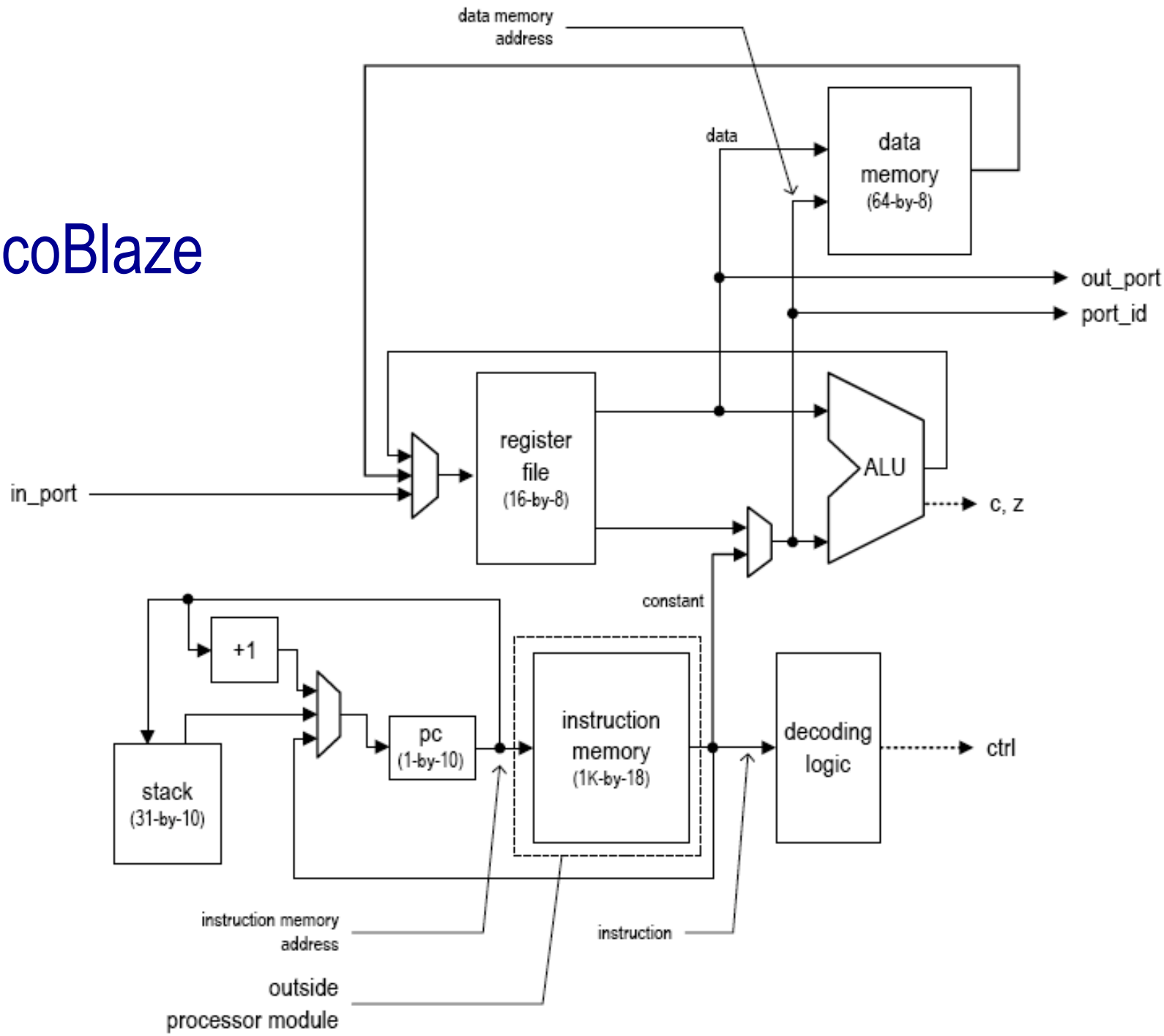
Block diagram of a Single-Purpose Processor (FSMD – Finite State Machine with Datapath)



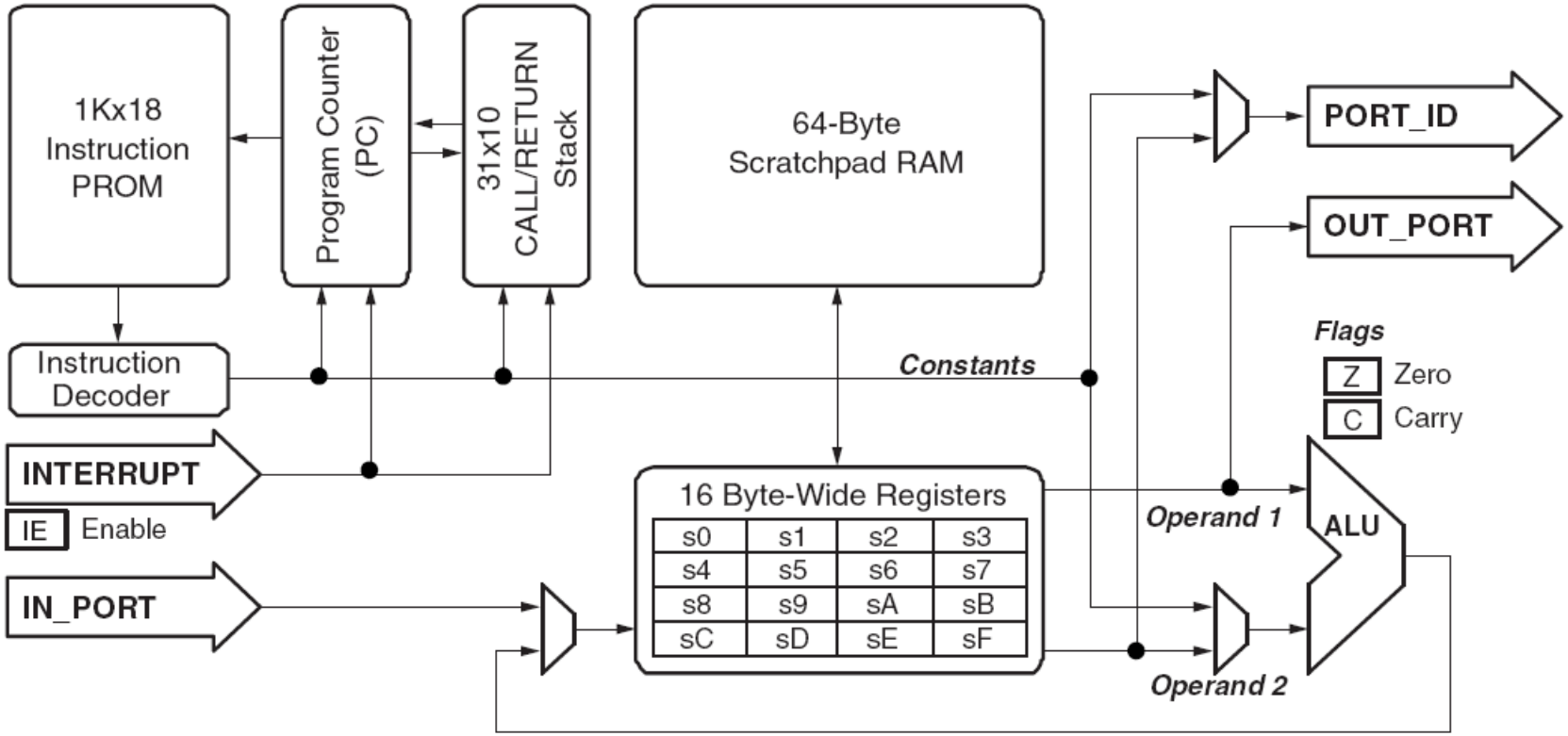
Block diagram of a General-Purpose Processor (Microcontroller)



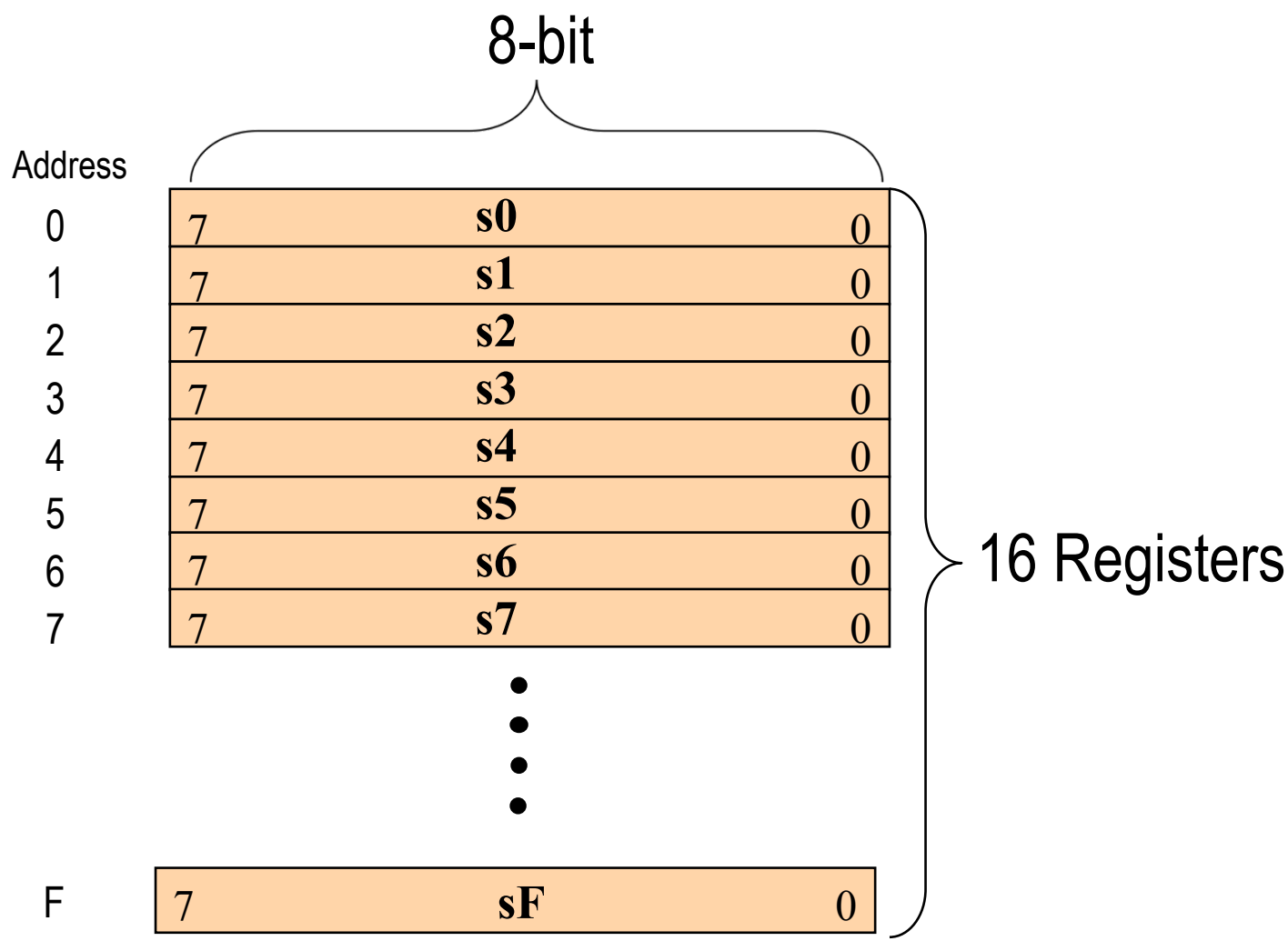
PicoBlaze


data memory address
data
data memory (64-by-8)
out_port
port_id
register file (16-by-8)
ALU
in_port
c, z
constant
+1
pc (1-by-10)
instruction memory (1K-by-18)
decoding logic
ctrl
stack (31-by-10)
instruction memory address
instruction
outside processor module

PicoBlaze Overview



Register File of PicoBlaze



Definition of Flags

Flags are set or reset after ALU operations

Zero flag - Z *zero condition*

Z = 1 if result = 0
0 otherwise

Carry flag - C *overflow, underflow, or various conditions*

<u>Example</u>*

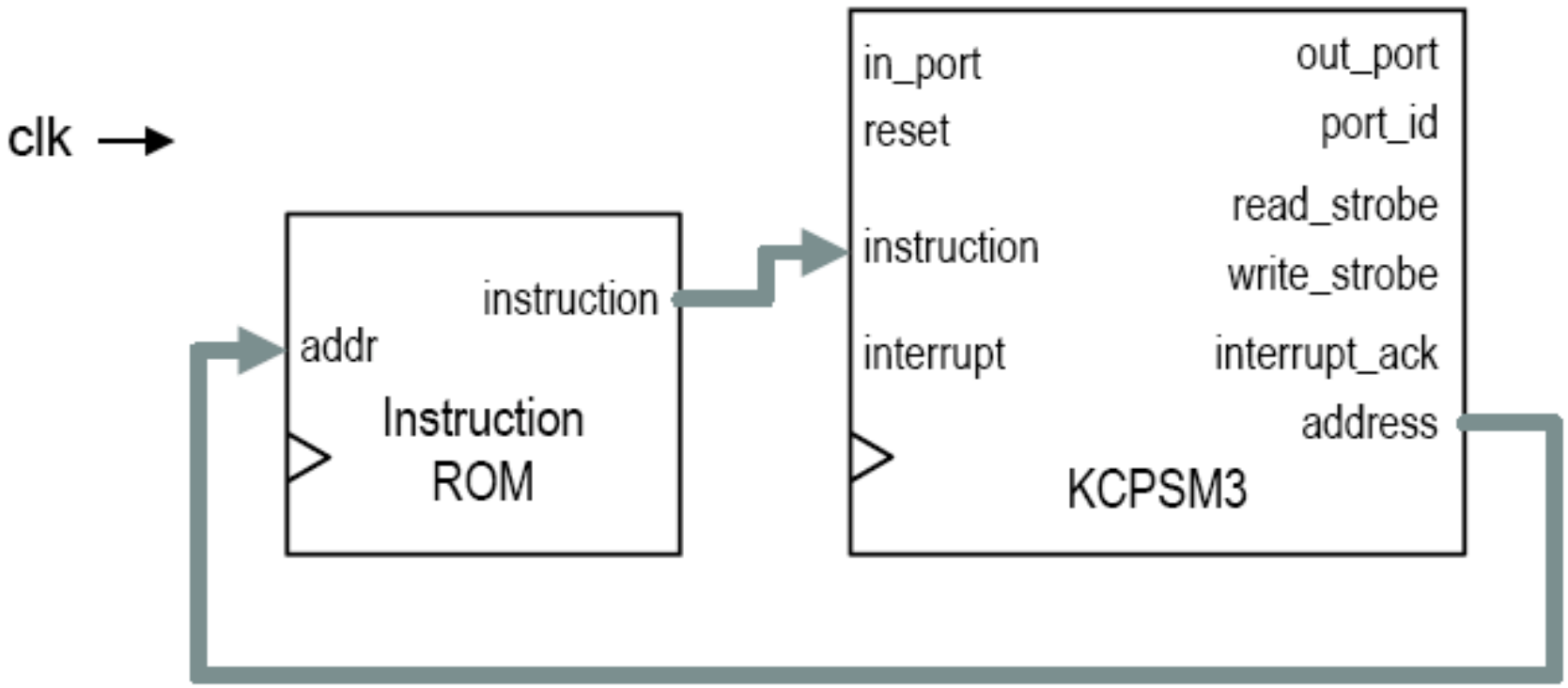
C = 1 if result > 2^8-1 **or**

result < 0

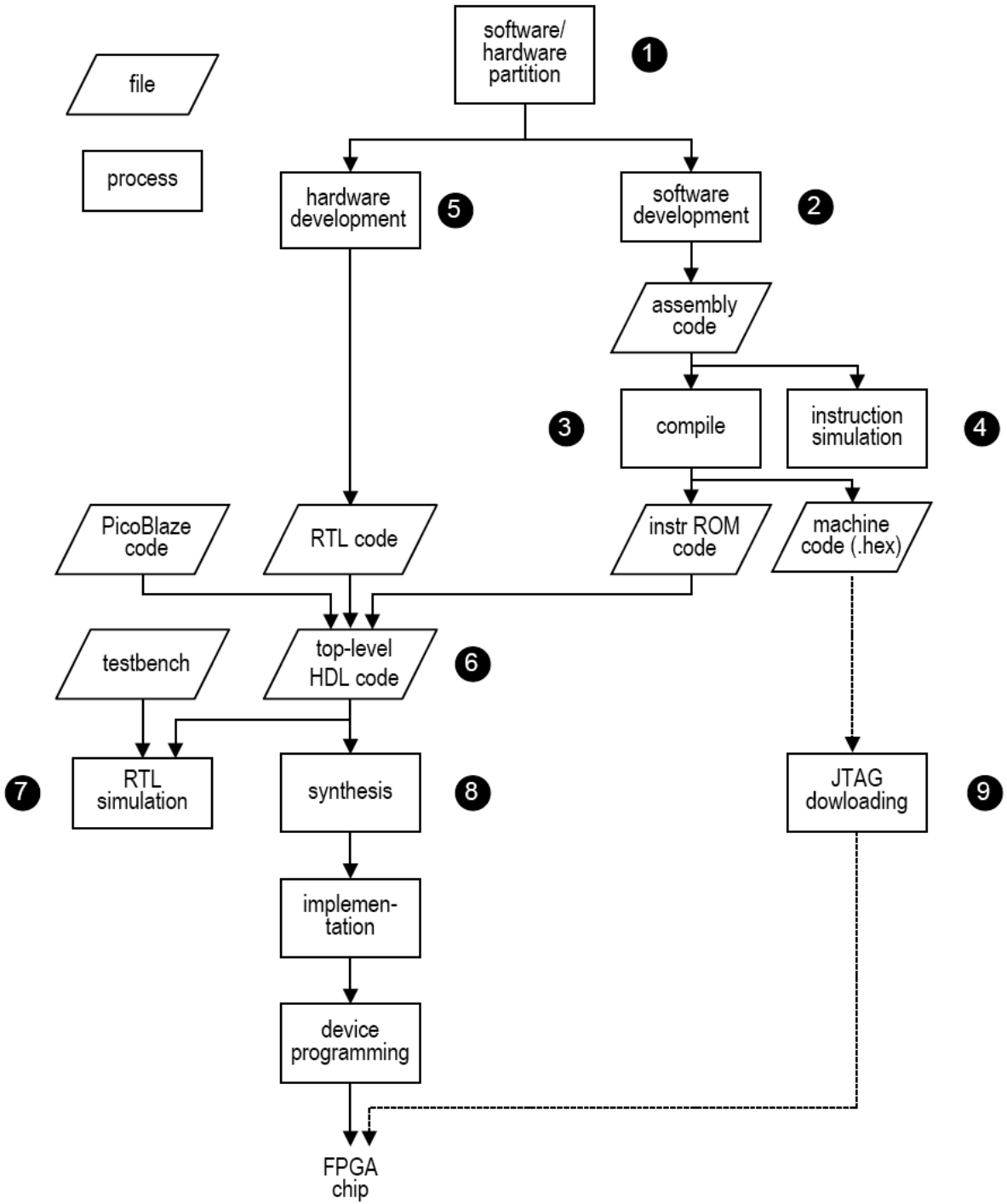
0 otherwise

***Applies only to addition or subtraction related instructions, refer to following slides otherwise**

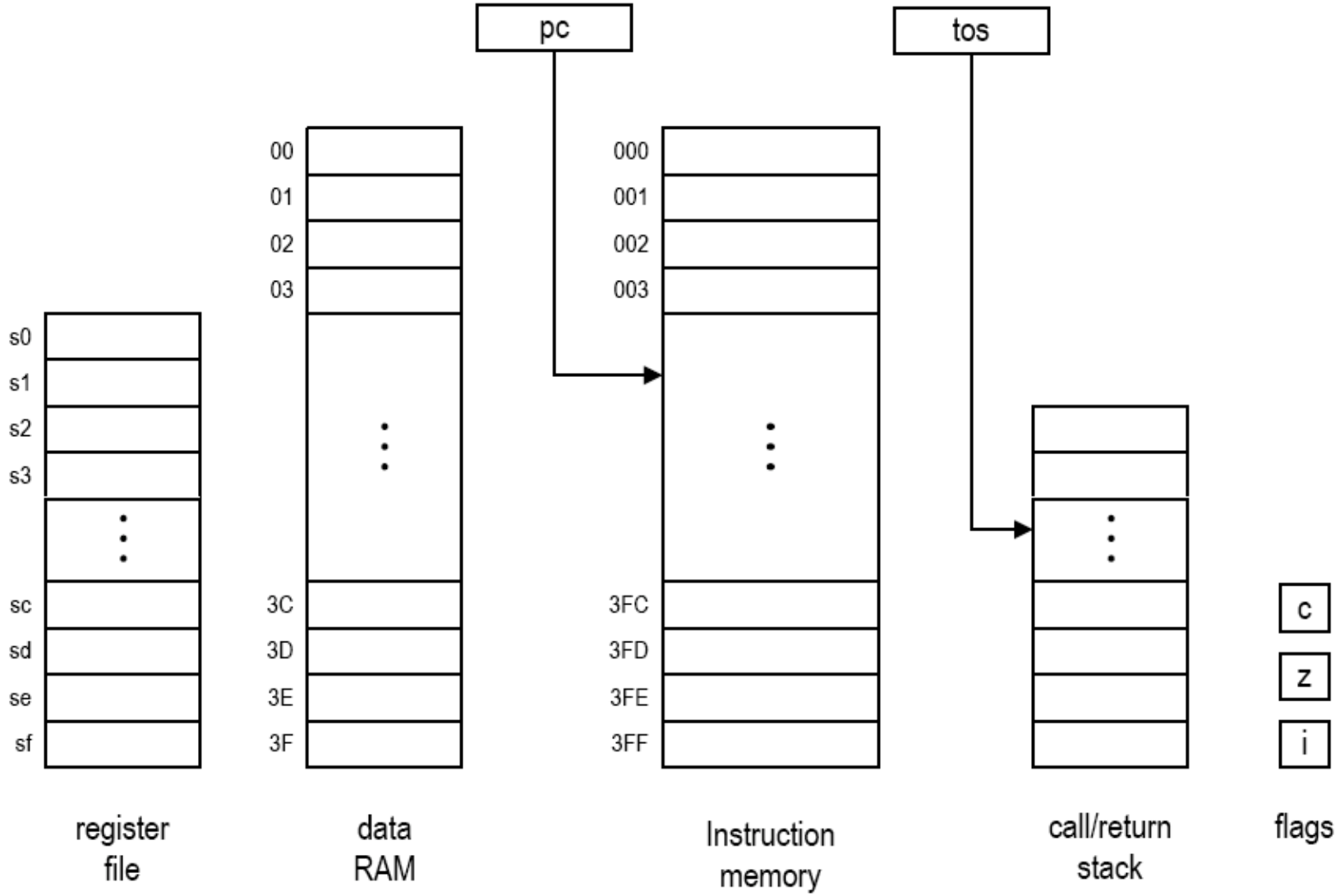
Interface of PicoBlaze



Development Flow of a System with PicoBlaze



PicoBlaze Programming Model



Syntax and Terminology

Syntax	Example	Definition
sX	s15	Value at register 15
KK	14	Value 14
PORT(KK)	PORT(2)	Input value from port 2
PORT((sX))	PORT((s10))	Input value from port specified by register 10
RAM(KK)	RAM(4)	Value from RAM location 4

Addressing modes

Immediate mode

ADDCY s2, 15	s2 + 15 + C → s2
SUB s7, 7	s7 – 7 → s7

Direct mode

INPUT s10, 28	PORT(28) → s10
ADD s10, s15	s10 + s15 → s10

Indirect mode

INPUT s9, s2	PORT((s2)) → s9
STORE s3, s10	s3 → RAM((s10))

PicoBlaze ALU Instruction Set Summary (1)

Instruction	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD sX,kk	0	1	1	0	0	0	x	x	x	x	k	k	k	k	k	k	k	k
ADD sX,sY	0	1	1	0	0	1	x	x	x	x	y	y	y	y	0	0	0	0
ADDCY sX,kk	0	1	1	0	1	0	x	x	x	x	k	k	k	k	k	k	k	k
ADDCY sX,sY	0	1	1	0	1	1	x	x	x	x	y	y	y	y	0	0	0	0
AND sX,kk	0	0	1	0	1	0	x	x	x	x	k	k	k	k	k	k	k	k
AND sX,sY	0	0	1	0	1	1	x	x	x	x	y	y	y	y	0	0	0	0
CALL	1	1	0	0	0	0	0	0	a	a	a	a	a	a	a	a	a	a
CALL C	1	1	0	0	0	1	1	0	a	a	a	a	a	a	a	a	a	a
CALL NC	1	1	0	0	0	1	1	1	a	a	a	a	a	a	a	a	a	a
CALL NZ	1	1	0	0	0	1	0	1	a	a	a	a	a	a	a	a	a	a
CALL Z	1	1	0	0	0	1	0	0	a	a	a	a	a	a	a	a	a	a
COMPARE sX,kk	0	1	0	1	0	0	x	x	x	x	k	k	k	k	k	k	k	k
COMPARE sX,sY	0	1	0	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
DISABLE INTERRUPT	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ENABLE INTERRUPT	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
FETCH sX, ss	0	0	0	1	1	0	x	x	x	x	0	0	s	s	s	s	s	s
FETCH sX,(sY)	0	0	0	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0
INPUT sX,(sY)	0	0	0	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
INPUT sX,pp	0	0	0	1	0	0	x	x	x	x	p	p	p	p	p	p	p	p

PicoBlaze ALU Instruction Set Summary (2)

JUMP	1	1	0	1	0	0	0	0	a	a	a	a	a	a	a	a	a	a
JUMP C	1	1	0	1	0	1	1	0	a	a	a	a	a	a	a	a	a	a
JUMP NC	1	1	0	1	0	1	1	1	a	a	a	a	a	a	a	a	a	a
JUMP NZ	1	1	0	1	0	1	0	1	a	a	a	a	a	a	a	a	a	a
JUMP Z	1	1	0	1	0	1	0	0	a	a	a	a	a	a	a	a	a	a
LOAD sX,kk	0	0	0	0	0	0	x	x	x	x	k	k	k	k	k	k	k	k
LOAD sX,sY	0	0	0	0	0	1	x	x	x	x	y	y	y	y	0	0	0	0
OR sX,kk	0	0	1	1	0	0	x	x	x	x	k	k	k	k	k	k	k	k
OR sX,sY	0	0	1	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
OUTPUT sX,(sY)	1	0	1	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
OUTPUT sX,pp	1	0	1	1	0	0	x	x	x	x	p	p	p	p	p	p	p	p
RETURN	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
RETURN C	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
RETURN NC	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
RETURN NZ	1	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0
RETURN Z	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
RETURNI DISABLE	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RETURNI ENABLE	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PicoBlaze ALU Instruction Set Summary (3)

RL sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	0	1	0
RR sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	1	0	0
SL0 sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	1	1	0
SL1 sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	1	1	1
SLA sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	0	0	0
SLX sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	1	0	0
SR0 sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	1	1	0
SR1 sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	1	1	1
SRA sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	0	0	0
SRX sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	0	1	0
STORE sX, ss	1	0	1	1	1	0	x	x	x	x	0	0	s	s	s	s	s	s
STORE sX,(sY)	1	0	1	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0
SUB sX,kk	0	1	1	1	0	0	x	x	x	x	k	k	k	k	k	k	k	k
SUB sX,sY	0	1	1	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
SUBCY sX,kk	0	1	1	1	1	0	x	x	x	x	k	k	k	k	k	k	k	k
SUBCY sX,sY	0	1	1	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0
TEST sX,kk	0	1	0	0	1	0	x	x	x	x	k	k	k	k	k	k	k	k
TEST sX,sY	0	1	0	0	1	1	x	x	x	x	y	y	y	y	0	0	0	0
XOR sX,kk	0	0	1	1	1	0	x	x	x	x	k	k	k	k	k	k	k	k
XOR sX,sY	0	0	1	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0

Logic instructions

		C	Z
1. AND AND sX, sY sX and sY => sX AND sX, KK sX and KK => sX	IMM, DIR	0	↕
2. OR OR sX, sY sX or sY => sX OR sX, KK sX or KK => sX	IMM, DIR	0	↕
3. XOR XOR sX, sY sX xor sY => sX XOR sX, KK sX xor KK => sX	IMM, DIR	0	↕

Arithmetic Instructions (1)

		C	Z
Addition	IMM, DIR	⬍	⬍
ADD sX, sY			
sX + sY => sX			
ADD sX, KK			
sX + KK => sX			
ADDCY sX, sY			
sX + sY + CARRY => sX			
ADDCY sX, KK			
sX + KK + CARRY => sX			

Arithmetic Instructions (2)

		C	Z
Subtraction	IMM, DIR	↕	↕
SUB sX, sY			
sX – sY => sX			
SUB sX, KK			
sX – KK => sX			
SUBCY sX, sY			
sX – sY – CARRY => sX			
SUBCY sX, KK			
sX – KK – CARRY => sX			

Test and Compare Instructions

		C Z
TEST	IMM, DIR	⬍⬍
TEST sX, sY		
sX and sY => none		**C = parity of the result**
TEST sX, KK		
sX and KK => none		
COMPARE	IMM, DIR	⬍⬍
COMPARE sX, sY		
sX – sY => none		
COMPARE sX, KK		
sX – KK => none		

Table 3-3: **COMPARE Instruction Flag Operations**

Flag	When Flag=0	When Flag=1
ZERO	Operand_1 ≠ Operand_2	Operand_1 = Operand_2
CARRY	Operand_1 ≥ Operand_2	Operand_1 < Operand_2

Data Movement Instructions (1)

		C	Z
LOAD	IMM, DIR	-	-
LOAD sX, sY			
sY => sX			
LOAD sX, KK			
KK => sX			

Data Movement Instructions (2)

		C	Z
STORE	DIR, IND	-	-

STORE sX, KK

sX => RAM(KK)

STORE sX, (sY)

sX => RAM((sY))

FETCH	DIR, IND	-	-

FETCH sX, KK

RAM(KK) => sX

FETCH sX, (sY)

RAM((sY)) => sX

Data Movement Instructions (3)

		C	Z
INPUT **INPUT sX, KK** **sX <= PORT(KK)** **INPUT sX, (sY)** **sX <= PORT((sY))**	DIR, IND	-	-
OUTPUT **OUTPUT sX, KK** **PORT(KK) <= sX** **OUTPUT sX, (sY)** **PORT((sY)) <= sX**	DIR, IND	-	-

Edit instructions - Shifts

Table 3-4: **PicoBlaze Shift Instructions**

	Shift Left ←		Shift Right →
SL0	Shift Left with '0' fill. CARRY Register sx 7 6 5 4 3 2 1 0 ← '0'	SR0	Shift Right with '0' fill. Register sx CARRY '0' → 7 6 5 4 3 2 1 0 →
SL1	Shift Left with '1' fill. CARRY Register sx 7 6 5 4 3 2 1 0 ← '1'	SR1	Shift Right with '1' fill. Register sx CARRY '1' → 7 6 5 4 3 2 1 0 →
SLX	Shift Left, eXtend bit 0. CARRY Register sx 7 6 5 4 3 2 1 0	SRX	Shift Right, sign eXtend. Register sx CARRY 7 6 5 4 3 2 1 0 →
SLA	Shift Left through All bits, including CARRY. CARRY Register sx 7 6 5 4 3 2 1 0	SRA	Shift Right through All bits, including CARRY. Register sx CARRY 7 6 5 4 3 2 1 0 →

*All shift instructions affect Zero and Carry flags

Edit instructions - Rotations

Table 3-5: **PicoBlaze Rotate Instructions**

	Rotate Left		Rotate Right
RL	CARRY, Register sx: 7 6 5 4 3 2 1 0	RR	Register sx: 7 6 5 4 3 2 1 0, CARRY

*All rotate instructions affect Zero and Carry flags

Program Flow Control Instructions (1)

JUMP AAA

PC <= AAA

JUMP C, AAA

if C=1 then PC <= AAA else PC <= PC + 1

JUMP NC, AAA

if C=0 then PC <= AAA else PC <= PC + 1

JUMP Z, AAA

if Z=1 then PC <= AAA else PC <= PC + 1

JUMP NC, AAA

if Z=0 then PC <= AAA else PC <= PC + 1

Program Flow Control Instructions (2)

CALL AAA

TOS <= TOS+1; STACK[TOS] <= PC; PC <= AAA

CALL C | Z , AAA

```
if C | Z =1 then
    TOS <= TOS+1; STACK[TOS] <= PC; PC <= AAA
else
    PC <= PC + 1
```

CALL NC | NZ , AAA

```
if C | Z =0 then
    TOS <= TOS+1; STACK[TOS] <= PC; PC <= AAA
else
    PC <= PC + 1
```

Program Flow Control Instructions (3)

RETURN

```
PC <= STACK[TOS]  + 1;  TOS <= TOS - 1
```

RETURN C | Z

```
if  C | Z =1  then
    PC <= STACK[TOS]  + 1;  TOS <= TOS - 1
else
    PC <= PC + 1
```

RETURN NC | NZ

```
if  C | Z =0  then
    PC <= STACK[TOS]  + 1;  TOS <= TOS - 1
else
    PC <= PC + 1
```

Interrupt Related Instructions

RETURNI ENABLE

PC <= STACK[TOS] ; TOS <= TOS – 1;

I <= 1; C<= PRESERVED C; Z<= PRESERVED Z

RETURNI DISABLE

PC <= STACK[TOS] ; TOS <= TOS – 1;

I <= 0; C<= PRESERVED C; Z<= PRESERVED Z

ENABLE INTERRUPT

I <=1;

DISABLE INTERRUPT

I <=0;