SGPC Programmer's Manual

Steven Vroom

November 2016

Contents

List of Figures

List of Tables

About This Manual

Related Documentation

Organization

Conventions

Acronyms and Abbreviations

Chapter 1

Overview

1.1 SGPC Architecture Overview

1.2 Registers

User-accessible Registers

Internal Registers

1.3 Instruction Conventions

Instruction Layout

Addressing Modes

1.4 Instruction Set

1.5 Interrupt Model

1.6 Memory Management Model

Chapter 2

Register Set

This chapter describes the registers seperated in four groups based on accessability. however, the internal registers are omitted from this chapter since these are implementation specific.

2.1 Foreground Registers

The foreground registers are the registers all regular instructions can read from and write to. There are eight 8-bit and eight 16-bit foreground registers. These registers are preserved in interrupts.

Table 2.1: List of Foreground Registers

ID	Mnonic	Descriptive Name	Length in bits
0x0	al	The lower byte of ax	8
0x1	ah	The higher byte of ax	8
0x2	bl	The lower byte of bx	8
0x3	bh	The higher byte of bx	8
0x4	cl	The lower byte of cx	8
0x5	ch	The higher byte of cx	8
0x6	dl	The lower byte of dx	8
0x7	dh	The higher byte of dx	8
0x8	ax	The first GPR	16
0x9	bx	The second GPR	16
0xA	cx	The third GPR	16
0xB	dx	The fourth GPR	16
0xC	ex	The fifth GPR	16
0xD	tm	Temporary data	16
0xE	sp	Stack pointer	16
0xF	pc	Program counter	16

General-Purpose Registers (GPRs)

These registers are meant for computing storage. The first four 16-bit registers are all splitted into two 8-bit registers. So software can directly access the upper and lower byte of these 16-bit registers.

Temporary Data Register

This registers is meant to facilitate call procedures. So it won't be preserved in a function call. However this nothing more than a suggestion to the user, software can use this register as a regular GPR.

Stack Pointer Register (SP)

This register is meant to keep track of the end of the stack. However this nothing more than a suggestion to the user, software can use this register as a regular GPR.

Program Counter Register (PC)

This register holds the address of the next instruction to run. Writing to this registers means jumping to other code.

2.2 Background Registers

Background registers can only accessed with the instructions BSTR and BLD.

Table 2.2: List of Background Registers

ID	Mnonic	Descriptive Name	Length in bits
0x0	n/a	Reserved	8
0x1	n/a	Reserved	8
0x2	n/a	Reserved	8
0x3	n/a	Reserved	8
0x4	n/a	Reserved	8
0x5	n/a	Reserved	8
0x6	n/a	Reserved	8
0x7	n/a	Reserved	8
0x8	bpc	Program counter backup	16
0x9	bsf	Segments and flags backup	16
0xA	ipc	Interrupt program counter	16
0xB	is	Interrupt segments	16
0xC	n/a	Reserved	16
0xD	n/a	Reserved	16
0xE	n/a	Reserved	16
0xF	n/a	Reserved	16

Backup Registers

The backup registers are used to backup the state of the CPU (see section 6.2). The foreground registers, segment registers and flags register all have their own backup register. However, most backup register aren't background registers. Only the segment registers, program counter register and flags register have directly accessable backup registers. Note that both segment registers and the flag register share one 16-bit backup register.

Interrupt Registers

The Interrupt registers hold the new state the CPU should jump to when an interrupt is triggered (see chapter 6). Only the segement registers and the program counter have an interrupt register.

2.3 Indirect Registers

Indirect registers are registers that software can't directly read nor write to with any instruction. All the backup registers that aren't background registers fall under this catogory. All the indirect registers can only be written to and read from via the state preservation system (see section 6.2).

Flags Register

The flags register holds multiple flags (see Table 2.3). There are two types of flags: status flags and control flags. Status flags give software extra information about the last computation made, while control flags control how the cpu behaves. There is only one control flag in the [insert name here] processor, the interrupts enabled flag. If this flag is set to zero interrupt requests will be ignored (see section 6.1).

Zero flag : If the result of the last computation is equal to zero this flag will be set, otherwise it will be cleared.

Sign flag : If the most significant bit of the result of the last computation is set this flag will be set, otherwise it will be cleared.

Parity flag : If the least significant bit of the result of the last computation is clear this flag will be set, otherwise it will be cleared.

Overflow flag : If the signed two's-complement result of the last computation is too large to fit in operand A (see section 3.1) this flag will be set, otherwise it will be cleared. (Not all computational instructions change this flag)

Carry flag : If the unsigned result of the last computation is too large to fit in operand A (see section 3.1) this flag will be set, otherwise it will be cleared. (Not all computational instructions change this flag)

Table 2.3: List of Flags

Mnonic	Descriptive Name	Type of flag	Length in bits
Z	Zero flag	Status flag	1
S	Sign flag	Status flag	1
P	Parity flag	Status flag	1
O	Overflow flag	Status flag	1
C	Carry flag	Status flag	1
I	Interrupts enabled flag	Control flag	1

Segment Registers

The segment registers (see Table 2.4) hold the index of the segments currently used (see chapter 5).

Table 2.4: List of Segment Registers

Mnonic	Descriptive Name	Length in bits
CS	Code segment index	5
DS	Data segment index	5

2.4 Output Registers

The output registers (see Table 2.5) hold information that is sent to the pheripherals (see chapter 7)

Table 2.5: List of Output Registers

Mnonic	Descriptive Name	Length in bits
AO	The first output register	16
BO	The second output register	16
CO	The third output register	16
DO	The fourth output register	16
EO	The fifth output register	16
FO	The sixth output register	16
GO	The seventh output register	16
HO	The eigth output register	16

Chapter 3

Operand Conventions and Addressing Modes

This chapter describes

3.1 Operand Conventions

Bit and Byte Ordering

Aligned and Misaligned Memory Access

3.2 Addressing Modes

Not From Memory

Register Direct

Absolute

Register with displacement

From Memory

Direct

Base Plus Displacement

Chapter 4

Instruction Set Summary

This chapter gives a summary on the instruction set of the [insert name here] processor. It will describe the types of instructions and the binary layout of the instructions.

4.1 Instruction Types

The instructions can be seperated in five types. Two of these types are not meant to be used.

Move Instructions

Arithimic Instructions

Control Instructions

Reserved Instructions

Artifact Instructions

Artifact Instructions are instructions with defined behaviour, but aren't considererd a permanent part of the [insert name here] processor family. These instructions tend to

4.2 Instruction Format

exceptions

Chapter 5

Memory Management

5.1 Segments

Base

Limit

5.2 Code Segment (CS)

5.3 Data Segment (DS)

5.4 Segment Switching

5.5 Changing Segments

Chapter 6

Interrupts

6.1 Interrupt Enabling/Disabling

Interrupt Enabled Flag

Internal Interrupt Mask

Programmable Interrupt Controller (PIC)

6.2 State Preservation

Backup

Full Restore

Partial Restore

6.3 Interrupt Service Routine (ISR)

Interrupt Far Jump

Return to Context

Switch Context

Chapter 7

I/O Conventions

7.1 Reading Input

7.2 Writing Output

Problem with Interrupts

Chapter 8

Instruction Set

MOVZ or MOV=

Opcode: 0x00

Usages

MOVZ a b
MOV= a b

Description

Writes B to A if the zero flag is set. Since the zero flag is always set after a comparison or test of two equal operators this instruction can also be used to only write A to B if the last comparison or test was equal. The alternative Memnonic MOV= was provided for this use.

Operation

if Z **then**
    $a \leftarrow b$
end if

alternative pseudocode:

if $y = 0$ **then** ▷ Where y is the result of a previous calculation
    $a \leftarrow b$
end if

Flags Affected

none

MOVNZ or MOV!=

Opcode: 0x01

Usages

MOVNZ a b
MOV!= a b

Description

Writes B to A if the zero flag is clear. Since the zero flag is always clear after a comparison or test of two unequal operators this instruction can also be used to only write A to B if the last comparison or test was unequal. The alternative Memnonic MOV= was provided for this use.

Operation

if $\neg Z$ **then**
  $a \leftarrow b$
end if

alternative pseudocode:

if $y \neq 0$ **then** ▷ Where y is the result of a previous calculation
  $a \leftarrow b$
end if

Flags Affected

none

0x02: MOVS

0x03: MOVNS

...

Appendix A

Instruction Set Listings

Table A.1: List of instructions sorted by opcode

Opcode			Memnonic	Operand A	Operand B
Decimal	Hex	Binary			
0	0x00	000000	MOVZ & MOV=	W?	R?
1	0x01	000001	MOVNZ & MOV!=	W?	R?
2	0x02	000010	MOVS	W?	R?
3	0x03	000011	MOVNS	W?	R?
4	0x04	000100	MOVP	W?	R?
5	0x05	000101	MOVNP	W?	R?
6	0x06	000110	MOVO	W?	R?
7	0x07	000111	MOVNO	W?	R?
8	0x08	001000	MOVC & MOVU>	W?	R?
9	0x09	001001	MOVNC & MOVU<=	W?	R?
10	0x0A	001010	MOVU>=	W?	R?
11	0x0B	001011	MOVU<	W?	R?
12	0x0C	001100	MOVS>	W?	R?
13	0x0D	001101	MOVS<=	W?	R?
14	0x0E	001110	MOVS<	W?	R?
15	0x0F	001111	MOVS>=	W?	R?
16	0x10	010000	MOV	W	R
17	0x11	010001	n/a	n/a	n/a
18	0x12	010010	WRI	_	R
19	0x13	010011	n/a	n/a	n/a
20	0x14	010100	STRB	O	R
21	0x15	010101	LDB	W	O
22	0x16	010110	OUT	O	R
23	0x17	010111	IN	W	O
24	0x18	011000	BACK	_	_
25	0x19	011001	FRET	_	_
26	0x1A	011010	PRET	_	_
27	0x1B	011011	FJMP	_	_
28	0x1C	011100	HLT	_	_
29	0x1D	011101	NOP	_	_
30	0x1E	011110	CMP	R	R

Table A.1: List of instructions sorted by opcode

Opcode			Memnonic	Operand A	Operand B
Decimal	**Hex**	**Binary**			
31	0x1F	011111	TEST	R	R
32	0x20	100000	OR	R&W	R
33	0x21	100001	OR	!R&W	R
34	0x22	100010	OR	R&W	!R
35	0x23	100011	OR	!R&W	!R
36	0x24	100100	AND	R&W	R
37	0x25	100101	AND	!R&W	R
38	0x26	100110	AND	R&W	!R
39	0x27	100111	AND	!R&W	!R
40	0x28	101000	XOR	R&W	R
41	0x29	101001	XOR	!R&W	R
42	0x2A	101010	XOR	R&W	!R
43	0x2B	101011	XOR	!R&W	!R
44	0x2C	101100	ADD	R&W	R
45	0x2D	101101	ADD	!R&W	R
46	0x2E	101110	ADD	R&W	!R
47	0x2F	101111	ADD	!R&W	!R
48	0x30	110000	ADD1	R&W	R
49	0x31	110001	ADD1	!R&W	R
50	0x32	110010	ADD1	R&W	!R
51	0x33	110011	ADD1	!R&W	!R
52	0x34	110100	ADDC	R&W	R
53	0x35	110101	ADDC	!R&W	R
54	0x36	110110	ADDC	R&W	!R
55	0x37	110111	ADDC	!R&W	!R
56	0x38	111000	SHL	W	R
57	0x39	111001	SHL1	W	R
58	0x3A	111010	RCL	W	R
59	0x3B	111011	ROL	W	R
60	0x3C	111100	SHR	W	R
61	0x3D	111101	SHR1	W	R
62	0x3E	111110	RCR	W	R
63	0x3F	111111	ROR	W	R

Appendix B

Simplified Mnemonics

Appendix C

Common Procedures

Appendix D

Standard Peripherals

D.1 Programmable Interrupt Controller (PIC)

D.2 Keyboard

D.3 Programmable Interrupt Timer (PIT)

D.4 Sound Card

D.5 Graphical Card

D.6 Memory Control Hub (MCH)

D.7 Segements and Out Of Bounds Exception