

65GZ032

PROGRAMMER'S REFERENCE GUIDE

Version 0.11
(5 May 2003)

Fabian Nunez
faybs@mad.scientist.com

CHANGELOG

Rev.	Date	Description
0.1	20 April 2003	Initial Release
0.11	5 May 2003	Change Request CR001 (branches); fix Table 2-7; some formatting changes

TABLE OF CONTENTS

Overview of the 65GZ032	1
65GZ032 Features	1
Full 32 Bit Design	1
Pipelined, Cached RISC Architecture	1
New Registers	1
New Opcodes	2
New Addressing modes	3
Paged Memory Management Unit	3
Instruction Set Reference	5
Detailed Opcode Descriptions	8
Mapping Of Instruction Bytes	8
ADD	9
ADDC	11
AND	13
BIT	15
BPL/BMI/BVC/BVS/BCC/BCS/BNE/BEQ/BGE/BLT/BGT/BLE/BPO/BRA	17
CFLAG	19
CMP	21
DEC	23
EOR	25
INC	27
JSR	29
LCNTR	31
LD	33
LMMU	35
MIRROR	37
MOV	39
MOVX	41
OR	43
RET	45
RETI	47
SCNTR	49
SFLAG	51
SHL	53
SHLC	55
SHR	57
SHRC	59
SMMU	61
ST	63
SUB	65
SUBC	67
SWAP	69
TRAP	71
TRAPB	73

OVERVIEW OF THE 65GZ032

The 6502 (and its variants, the 6510 and the 8502) was and is one of the most popular 8-bit CPUs. However, the fact that its design is over 20 years old has prompted some to modernize its design. The results of these improvements have been the 65C02 and the 65C816, which have speeded up the CPU and added some new instructions (and in the case of the 65C816, 16-bit registers). However, these improvements have tried to stay as close as possible to the original 6502 design, and so have been only moderately successful.

The 65GZ032 approaches the task of modernizing the 6502 architecture from a different angle: create a CPU that is binary-compatible with the 6502, while adding a set of modern RISC instructions that deviate significantly from the original design. This is a “best of both worlds” solution to the problem, as existing code (that does not use any undocumented opcodes) will work, while new code can take advantage of the powerful new features of this CPU.

65GZ032 FEATURES

As stated above, the 65GZ032 features full binary compatibility with the 6502. It also provides the following new features:

- Full 32 bit design
- Pipelined RISC architecture, with instruction and data caches
- Eight 32 bit registers (five are all-purpose) and one 16 bit register
- Several new opcodes
- New addressing modes
- Highly orthogonal architecture (very few fixed-function registers)
- Can access 4 GB of linear memory (no segments, bank switching, etc)
- Built-in paged memory management unit
- Much higher clock speeds (33MHz)

FULL 32 BIT DESIGN

The 65GZ032 is a 32 bit processor, with the ability to operate on 8, 16 and 32 bit wide operands. Its address and data buses are 32 bits wide, in contrast with the 6502, which has a 16-bit address bus and an 8-bit data bus. All the 6502 8-bit opcodes are translated by the 65GZ032 into one or two 32-bit opcodes during instruction fetch. Since the 65GZ032 is very different internally from the 6502, the instruction timings for many opcodes are different. However, because of the advanced design of the 65GZ032, the difference is usually in favour of the newer CPU.

PIPELINED, CACHED RISC ARCHITECTURE

The 65GZ032 is based on a RISC design, which emphasizes the use of small building block instructions, and many general-purpose registers that can be used orthogonally (e.g. LDX \$20,A is as valid as LDA \$20,X). By contrast, the 6502 is a CISC design, with a small number of dedicated-purpose registers (A, X, Y).

The 65GZ032 also features a pipelined architecture, which allows several instructions to be in different stages of execution simultaneously.

Finally, the 65GZ032 also contains high-speed cache memory for instructions and data. This allows the CPU to operate at a speed higher than the system bus normally allows.

NEW REGISTERS

The 6502 has 5 8-bit registers (A, X, Y, SP and Flags) and 1 16-bit register (PC). Each of these registers has a fixed function. The 65GZ032 has 8 32-bit registers (R0-R7) and 1 16-bit register (Flags). Table 1-1 has the complete list of available registers. These registers overlap with the 6502 register set as indicated in Table 1-2.

Table 1-1. 65GZ032 Registers

Register	Alt. Name	Description
R0	Z or Ri	Zero, or Implicit Register
R1	A	Accumulator, also general purpose register
R2	X	X offset register, also general purpose register
R3	Y	Y offset register, also general purpose register
R4	G	Gideon's Register (general purpose)
R5	J	Jeri's Register (general purpose)
R6	SP	Stack Pointer
R7	PC	Program Counter
Flags	Flags	CPU Flags

Table 1-2. Equivalent 65GZ032 registers for 6502 Registers

6502 Register	65GZ032 Register
A	R1
X	R2
Y	R3
SP	Least significant 8 bits of R6
PC	Least significant 16 bits of R7
Flags	Least significant 8 bits of Flags

THE INTERNAL 'IMPLICIT' REGISTER

Register 'R0' is a special register. Generally, it will always read 'zero' upon entering the first phase of a multi-phase instruction. However, when 'R0' is set as destination register, it will set an internal control bit that enables the temporary value to be read in the next instruction, or next phase of a multi-phase instruction. This enables the linkage of instructions by means of a temporary value. This bit is automatically reset. No interrupts can occur when this bit is set, so the next instruction will always be executed after the instruction that 'writes' R0. The latter provides a possibility to execute atomic operations in multithreaded operating systems. This 'implicit' register feature is used in sequences like:

```
SHR [R5+$1234], #3
ST.L R0, [SP--]
```

The first instruction shifts the long-word three positions to the right; the second instruction pushes this temporary value onto the stack.

NEW OPCODES

The 6502 has a large number of 'illegal' opcodes. These 'open spaces' in the opcode table can be used to extend the instruction set, since it can be reasonably expected that no programs will be using those opcodes. One characteristic of the 6502 instruction set is that all opcodes that have the least significant two bits set (i.e. all opcodes with binary representation xxx xx11) are illegal. This fact is taken advantage of by the 65GZ032, which uses all such opcodes as the first byte of its extended, 32-bit opcodes. These extended opcodes also make use of a second byte; therefore all extended opcodes are of the form xxx xx11 xxxx xxxx.

NEW ADDRESSING MODES

As well as supporting the known 6502 addressing modes, the 65GZ032 supports six new or updated addressing modes, as shown in Table 1-3.

Table 1-3. New/updated Addressing modes

Form	Description
[reg]	Register indirect (R0-R7)
[reg,offset16]	Register indirect with 16-bit offset
reg	Register (R0-R7)
imm16	16-bit immediate value
[reg++]	Register indirect with autoincrement
[--reg]	Register indirect with autoincrement

AUTO INCREMENT AND DECREMENT FUNCTION

The 65GZ032 supports auto increment and decrement of memory references. One use of this feature is to push values to the stack and pop them from the stack. Usually, incrementing is done after the use of the value, while decrementing is done before using the value. Unfortunately, the 6502 does this the other way around when accessing the stack. When a value is pushed, the stack pointer is used and then decremented. For this reason, this CPU supports both ways. Then the stack pointer is referenced, the order of incrementing or decrementing and using the value is reversed. In all other cases, the regular order is used. Table 1-4 depicts the valid combinations:

Table 1-4. Auto increment/decrement order

Size	Mode	First	Second
.B	[Rn++]	[Rn]	$R_n = R_n + 1$
.W	[Rn++]	[Rn]	$R_n = R_n + 2$
.L	[Rn++]	[Rn]	$R_n = R_n + 4$
.B	[--Rn]	$R_n = R_n - 1$	[Rn]
.W	[--Rn]	$R_n = R_n - 2$	[Rn]
.L	[--Rn]	$R_n = R_n - 4$	[Rn]
.B	[++SP]	$SP = SP + 1$	[SP]
.W	[++SP]	$SP = SP + 2$	[SP]
.L	[++SP]	$SP = SP + 4$	[SP]
.B	[SP--]	[SP]	$SP = SP - 1$
.W	[SP--]	[SP]	$SP = SP - 2$
.L	[SP--]	[SP]	$SP = SP - 4$

PAGED MEMORY MANAGEMENT UNIT

The 65GZ032 includes a Memory Management Unit (MMU). This unit makes it possible assign a virtual address to any physical memory address. This allows code or data to be moved to different locations in physical memory, while retaining their original addresses (e.g. the VIC registers could be mapped so that they appear to be at any virtual address, e.g. \$FFFFD000).

Another feature of the MMU is that it provides a way to specify to which memory-bus an access will be routed. This enables the use of older (and therefore slower) I/O devices, while at the same time being able to use faster memory for code execution.

The MMU works by mapping physical to virtual addresses in 1KB blocks of memory (this size corresponds to the smallest block of memory-space in the C-64 that contains a distinctive function, which might need a different MMU setting). The table that is used to control this mapping will be located in general external memory, like SDRAM. The MMU has a register that holds the base address for this translation table. The operating system can change this base address with every context switch, to give each running program the illusion that it has the entire 4G address space for itself, or to maintain different per-process memory mappings.

Since every memory access needs to be translated, in order to prevent every single memory access from becoming two (one for the MMU table and another for the actual access), the CPU holds the most recently used translation entries in a cache called the TLB (Translation Lookaside Buffer).

As an example, Table 1-5 illustrates a possible configuration for the C64's I/O address range.

Table 1-5. Possible MMU Configuration

Physical Memory Range	Virtual Memory Range	Function	MMU Setting
D000 – D3FF	FFFF0000 – FFFF03FF	VIC	R/W routed to I/O bus
D400 – D7FF	FFFF0400 – FFFF07FF	SID	R/W routed to I/O bus
D800 – DBFF	FFFF0800 – FFFF0BFF	Color RAM	Write to both buses Read from SDRAM
DC00 – DFFF	FFFF0C00 – FFFF0DFF	I/O	R/W Routed to I/O bus

To ensure C-64 compatibility, the MMU will be disabled on start-up; this means that all memory access will be routed to the I/O bus. System software can then enable the MMU and copy the ROMs into SDRAM and program the MMU look up tables such that the original ROMs seem to appear at the normal locations (this will make the code in the ROMS appear to be at the same location, but the code will in reality be in fast SDRAM, increasing performance).

Any change in the signals that control the presence of the C64 ROMs (i.e., writes to memory locations \$0000 and \$0001) will generate an MMU interrupt, so system software can correctly update the MMU look up tables and thus correctly emulate C-64 functionality. This could be extended to include the VIC bank select signals, in order to improve performance for those areas that the VIC is not working in (one should use this feature with care, since there are programs that quickly switch between banks!).

It is also possible to generate a trap when a given block of memory is accessed. This enables the use of the MMU for virtual memory management, but it can also be used for emulation purposes. For example, when a trap is set to be generated on the area DC00 – DFFF, various expansion cartridges such as the REU can be emulated in software.

INSTRUCTION SET REFERENCE

The following section documents only the new opcodes introduced by the 65GZ032 CPU, i.e. those that are not present in the original 6502. Table 2-1 lists the opcode formats for all instructions, where:

- 'm' stands for memory access mode bit (see Tables 2-2 and 2-3)
- 'R' stands for register select bit (000b–111b for R0-R7)
- 'S' stands for source register select bit (000b–111b for R0-R7)
- 'D' stands for destination register select bit (000b–111b for R0-R7)
- 'w' stands for data width control (see Tables 2-4 and 2-5)
- 'c' stands for carry on/off
- '#' stands for a value parameter
- 'F' stands for flag selection bit (see Table 2-6)
- 'b' stands for break flag on/off
- 'e' stands for sign extension on/off
- 'x' stands for a don't care bit (not used)
- 'B' stands for branch test select (see Table 2-7)
- 'd' stands for direction ('0' means written, '1' means read)

Table 2-1. The 14-bit instruction format

Opcode	Mnemonic	Description	Format
0000	OR	Logical 'or'	0000 mm SSS DDD ww
0001	AND	Logical 'and'	0001 mm SSS DDD ww
0010	EOR	Logical 'exclusive or'	0010 mm SSS DDD ww
0011	ADD	Add	0011 mm SSS DDD ww
0100	ADDC	Add with carry	0100 mm SSS DDD ww
0101	SUB	Subtract	0101 mm SSS DDD ww
0110	SUBC	Subtract with carry	0110 mm SSS DDD ww
0111	CMP	Compare	0111 mm SSS DDD ww
1000	LD	Load from memory	1000 mm SSS DDD ww
1001	ST	Store to memory	1001 mm DDD SSS ww
1010	MOV	Move registers	1010 em SSS DDD ww
1011 000	JSR	Jump to subroutine	1011 mm RRR 000 xw
1011 001	RET	Return from subroutine	1011 xx xxx 001 xw
1011 010	TRAP	Trap (vectored)	1011 ## ### 010 bw
1011 011	RETI	Return from interrupt	1011 xx xxx 011 xw
1011 100	CNTR	Access control register	1011 mm RRR 100 dw
1011 101	FLAG	Set/Clear flags	1011 #F FFF 101 xx
1011 110	MMU	Access MMU	1011 mm RRR 110 d#
1011 111	Expansion		1011 xx xxx 111 xx
1100	SHL	Shift left	1100 mm SSS c## ww
1101	SHR	Shift right	1101 mm SSS c## ww
1110 000	INC	Increment	1110 mm RRR 000 ww
1110 001	DEC	Decrement	1110 mm RRR 001 ww
1110 010	SWAP	Swap bytes/words	1110 mm RRR 010 ww
1110 011	MIRROR	Mirror bits	1110 mm RRR 011 ww
1110 100	BIT	Bit test	1110 mm RRR 100 ww
1110 101	Expansion		1110 xx xxx 101 xx
1110 110	Expansion		1110 xx xxx 101 xx
1110 111	Expansion		1110 xx xxx 101 xx
1111	BR	Branch	1111 BB BB# ### ##

Table 2-2. 2 Bit Addressing Mode Bit Encodings

Encoding	ALU Operations		Load/Store Operations	
00	[reg]	Register Indirect	[reg]	Register Indirect
01	[reg,offset16]	Register Indirect with Offset	[reg,offset16]	Register Indirect with Offset
10	reg	Register	[reg++]	Register Indirect with Autoincrement
11	imm16	Immediate	[--reg]	Register Indirect with Autodecrement

NOTE: The autoincrement/autodecrement addressing modes add or subtract the instruction's data width from the base register, i.e. opcode.L adds or subtracts 4, opcode.W and opcode.H 2, and opcode.B 1.

NOTE: If the autoincrement/autodecrement addressing modes are used and the destination register is SP (R6), the order is backwards: for autoincrement, the increment happens first, and for autodecrement, the decrement happens last. This is for backwards compatibility with the 6502's PUSH and POP opcodes.

NOTE: For all instructions that have a 32 bit addressing mode (opcode.L), but the operand is a 16 bit immediate, the upper 16 bits of the result will be a copy of the upper 16 bits of R0.

For example, `MOVE.W #$1234,R0H; MOVE.L #$5678,R2` will implicitly copy the upper 16 bits of R0 into the upper 16 bits of R2, with the end result being that R2 ends up with the value \$12345678; Additionally, this operation will be performed atomically (because of the write to R0).

Table 2-3. 1 bit Addressing Mode Bit Encodings

Encoding	Addressing Mode
0	reg
1	imm16

Table 2-4. 2 bit Width Selection Bit Encodings

Encoding	Width	Assembler Syntax
00	8 bits	opcode.B ..., Rn
01	16 bits (low word)	opcode.W ..., Rn opcode.W ..., RnL
10	16 bits (high word)	opcode.H ..., Rn opcode.W ..., RnH
11	32 bits	opcode.L ..., Rn

Table 2-5. 1 bit Width Selection Bit Encodings

Encoding	Width (JSR, RET, RETI, TRAP)	Width (all other opcodes)
0	16 bits	8 bits
1	32 bits	16 bits

Table 2-6. Flag Selection Bit Encodings

Encoding	Abbrev	Name	Encoding	Abbrev	Name
0000	C	Carry	1000	–	–
0001	Z	Zero	1001	–	–
0010	I	Interrupt Disable	1010	FDC	Flush Data Cache
0011	D	Decimal	1011	FIC	Flush Inst. Cache
0100	B	Break	1100	DCA	Data Cache Enabled
0101	E	(Emulation?)	1101	ICA	Inst. Cache Enabled
0110	V	oVerflow	1110	FTL	Flush Translation Lookup
0111	N	Negative	1111	MMU	MMU Enabled

Table 2-7. Branch Test Bit Encodings

Encoding	Mnemonic	Test	Description
0000	BPL	not(N)	Positive or Zero
0001	BMI	N	Negative
0010	BVC	not(V)	Overflow cleared
0011	BVS	V	Overflow set
0100	BCC	not(C)	Carry cleared
0101	BCS	C	Carry set
0110	BNE	not(Z)	Not equal / zero
0111	BEQ	Z	Equal or zero
1000	BGE	not (N xor V)	Greater or Equal
1001	BLT	(N xor V)	Less than
1010	BGT	not(Z or (N xor V))	Greater than
1011	BLE	Z or (N xor V)	Less or Equal
1100	BPO	not(N or Z)	Positive
1101	--	--	Future Expansion
1110	BRA	'1'	Always
1111	--	--	Future Expansion

DETAILED OPCODE DESCRIPTIONS

In the following pages, each of the possible opcodes is fully described. Note that at this moment only the RISC syntax is described; a full CISC syntax description will follow later.

MAPPING OF INSTRUCTION BYTES

Table 1-4 shows the format of each instruction in a 14-bit word. These instructions map on the byte-addressable memory as follows:

Byte 0								Byte 1								Byte 2		Byte 3	
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0				
13	12	11	10	9	8	'1'	'1'	7	6	5	4	3	2	1	0	Operand LSB		Operand MSB	

Example: SUBC

0	1	1	0	m	M	1	1	S	S	S	D	D	D	w	w		
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--	--

ADD

Summary: Binary Addition (no carry)
Operation: Register/Memory + Register → Register
Assembler Syntax: ADD SrcOp, Rn

Description: Adds the source and destination operands and stores the result in the destination register. The size of the operation may be specified as byte, word (high or low) or long. Note that this ignores the carry bit (see ADDC).

Affected Flags:

C	Z	I	D	B	E	V	N
✓	✓	–	–	–	–	✓	✓

C — Set if a carry is generated; cleared otherwise.
 Z — Set if the result is zero; cleared otherwise.
 V — Set if an overflow is generated; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	1	1	Addr Mode	1	1		Source Register			Destination Register			Data Width	
Immediate/offset LSB (only if used by addressing mode)								Immediate/offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	ADD [reg],reg	Register Indirect
01	ADD [reg,offset16],reg	Register Indirect with Offset
10	ADD reg,reg	Register
11	ADD imm16,reg	Immediate

Source/Destination Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	ADD.B SrcOp,Rn	8 bits
01	ADD.W SrcOp,Rn ADD.W SrcOp,RnL	16 bits (low word)
10	ADD.H SrcOp,Rn ADD.W SrcOp,RnH	16 bits (high word)
11	ADD.L SrcOp,Rn	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the 32 bit form (.L) is used, but the source operand is a 16 bit constant, the upper 16 bits copied into the destination operand will be the upper 16 bits of R0.

ADDC

Summary: Binary Addition (with carry)
Operation: Register/Memory + Register + Carry → Register
Assembler Syntax: ADDC SrcOp, Rn

Description: Adds the source and destination operands, plus the carry bit of the Flags register, and stores the result in the destination register. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
✓	✓	–	–	–	–	✓	✓

C — Set if a carry is generated; cleared otherwise.
 Z — Set if the result is zero; cleared otherwise.
 V — Set if an overflow is generated; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	1	0	0	Addr Mode	1	1		Source Register			Destination Register			Data Width	
Immediate/offset LSB (only if used by addressing mode)								Immediate/offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	ADDC [reg],reg	Register Indirect
01	ADDC [reg,offset16],reg	Register Indirect with Offset
10	ADDC reg,reg	Register
11	ADDC imm16,reg	Immediate

Source/Destination Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	ADDC. B SrcOp,Rn	8 bits
01	ADDC. W SrcOp,Rn ADDC. W SrcOp,Rn L	16 bits (low word)
10	ADDC. H SrcOp,Rn ADDC. W SrcOp,Rn H	16 bits (high word)
11	ADDC. L SrcOp,Rn	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the 32 bit form (.L) is used, but the source operand is a 16 bit constant, the upper 16 bits copied into the destination operand will be the upper 16 bits of R0.

AND

Summary: Bitwise AND
Operation: Register/Memory \wedge Register \rightarrow Register
Assembler Syntax: AND SrcOp, Rn

Description: Performs a bitwise “AND” of the source operand and the destination operand and stores the result in the destination register. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
–	✓	–	–	–	–	–	✓

Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	1	Addr Mode	1	1		Source Register			Destination Register			Data Width	
Immediate/offset LSB (only if used by addressing mode)								Immediate/offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	AND [reg],reg	Register Indirect
01	AND [reg,offset16],reg	Register Indirect with Offset
10	AND reg,reg	Register
11	AND imm16,reg	Immediate

Source/Destination Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon’s Register (general purpose)
101	R5	J	Jeri’s Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	AND. B SrcOp,Rn	8 bits
01	AND. W SrcOp,Rn AND. W SrcOp,Rn L	16 bits (low word)
10	AND. H SrcOp,Rn AND. W SrcOp,Rn H	16 bits (high word)
11	AND. L SrcOp,Rn	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the 32 bit form (.L) is used, but the source operand is a 16 bit constant, the upper 16 bits copied into the destination operand will be the upper 16 bits of R0.

BIT

Summary: Test bits against R1 (Accumulator)

Operation: $\text{Not}(\text{Register/Memory} \wedge R1) \rightarrow Z, (\text{Register/Memory})[7] \rightarrow N, (\text{Register/Memory})[6] \rightarrow V$

Assembler Syntax: BIT SrcOp

Description: Tests bits against the accumulator (R1). Bits 6 and 7 of the source operand are transferred to the overflow and negative flags of the status register, respectively. R1 is ANDed with the source operand, and if the result is zero, the zero flag of the status register is set; else it is cleared. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
-	✓	-	-	-	-	✓	✓

Z — Set if the source operand AND R1 is zero; cleared otherwise.

V — Copied from bit 6 of the source operand.

N — Copied from bit 7 of the source operand.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	1	1	0	Addr Mode	1	1		Source Register	1	0	0			Data Width	
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	BIT [reg]	Register Indirect
01	BIT [reg,offset]6	Register Indirect with Offset
10	BIT reg	Register

Source Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	BIT. B SrcOp	8 bits
01	BIT. W SrcOp	16 bits (low word)
10	BIT. H SrcOp	16 bits (high word)
11	BIT. L SrcOp	32 bits

Notes:

1. This opcode operates the same as the 6502 BIT opcode.

BPL/BMI/BVC/BVS/BCC/BCS/BNE/BEQ/BGE/BLT/BGT/BLE/BPO/BRA

Summary: Branch on condition
Operation: Branch on condition
Assembler Syntax: Bcc destination

Description: If the specified condition is true, program execution continues at location (PC) + displacement. The program counter contains the address of the instruction word for the Bcc instruction plus two. The displacement is a two's-complement integer that represents the relative distance in bytes from the current program counter to the destination program counter. This displacement is 20 bits wide, which translates into half a megabyte in either direction. If you wish to use a shorter branch, the 6502-compatible 8-bit branch opcodes are still available, their assembler syntax is **Bxx.B dest** (e.g. **BCC.B out**, **BNE.B again**).

The 'Taken' bit is used to provide a hint to the CPU pipeline. If this bit is '0', then the branch is expected to *not* be taken most of the time (e.g. the 'exit' test for a loop); otherwise it is expected to *be* taken most of the time (e.g. the 'loop back' test for a loop).

Affected Flags:

C	Z	I	D	B	E	V	N
-	-	-	-	-	-	-	-

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	1	1	1	Test (bits 1-0)	1	1	Test (bits 3-2)	Taken	1	Branch Displ. (bits 19-16)					
Branch Displacement (bits 7-0)								Branch Displacement (bits 15-8)							

Branch Condition Tests:

Encoding	Assembler Syntax	Test	Description
0000	BPL	not(N)	Positive or Zero
0001	BMI	N	Negative
0010	BVC	not(V)	Overflow clear
0011	BVS	V	Overflow set
0100	BCC	not(C)	Carry cleared
0101	BCS	C	Carry set
0110	BNE	not(Z)	Not equal / zero
0111	BEQ	Z	Equal or zero
1000	BGE	not (N xor V)	Greater or Equal
1001	BLT	(N xor V)	Less than
1010	BGT	not(Z or (N xor V))	Greater than
1011	BLE	Z or (N xor V)	Less or Equal
1100	BPO	not(N or Z)	Positive
1101	--	--	Future Expansion
1110	BRA	'1'	Always
1111	--	--	Future Expansion

Taken Bit:

Encoding	Assembler Syntax	Branch Taken?
0	Bxx.N dest	Expect not
1	Bxx.T dest	Expect yes

Notes:

1. BRA makes code position-independent, but is limited to a 20-bit offset. For longer jumps use ADD Rn,R7
2. Bit 4 of the second byte of the instruction is the Extend bit. If this is set to one, the offset is 20 bits wide, else it is 16 bits wide and the least significant 4 bits of that byte are ignored (implied 0). This is used by the internal 6502 to 65GZ032 opcode translator; mere humans must set this bit to '1'.
3. The Branch Taken bit can noticeably improve the performance of your code – use it as often as you can!
4. 6502-compatible 8-bit branches cannot have their Taken bits explicitly set.

CFLAG

Summary: Clear one bit in the Flags register
Operation: 0 → Flags[k]
Assembler Syntax: CFLAG imm4

Description: Sets any bit in the Flags register to '0'.

Affected Flags:

C	Z	I	D	B	E	V	N	–	–	FDC	FIC	DCA	ICA	FTL	MMU
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

You can clear any flag bit.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	1	1	0	Flag (bit 0)	1	1	Flag (bits 3–1)			1	0	1	–	–

Flag Number:

Encoding	Abbrev	Name	Encoding	Abbrev	Name
0000	C	Carry	1000	–	–
0001	Z	Zero	1001	–	–
0010	I	Interrupt Disable	1010	FDC	Flush Data Cache
0011	D	Decimal	1011	FIC	Flush Inst. Cache
0100	B	Break	1100	DCA	Data Cache Enabled
0101	E	(Emulation?)	1101	ICA	Inst. Cache Enabled
0110	V	oVerflow	1110	FTL	Flush Translation Lookup
0111	N	Negative	1111	MMU	MMU Enabled

(this page intentionally left blank)

CMP

Summary: Compare
Operation: Register/Memory – Register
Assembler Syntax: CMP SrcOp, Rn

Description: Subtracts the destination from the source operands, but does not store the result anywhere. This effectively compares the two values. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
✓	✓	–	–	–	–	–	✓

C — Set if a carry is generated; cleared otherwise.
 Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	1	1	1	Addr Mode	1	1		Source Register			Destination Register			Data Width	
Immediate/offset LSB (only if used by addressing mode)								Immediate/offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	CMP [reg],reg	Register Indirect
01	CMP [reg,offset16],reg	Register Indirect with Offset
10	CMP reg,reg	Register
11	CMP imm16,reg	Immediate

Source/Destination Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	CMP. B SrcOp,Rn	8 bits
01	CMP. W SrcOp,Rn CMP. W SrcOp,Rn L	16 bits (low word)
10	CMP. H SrcOp,Rn CMP. W SrcOp,Rn H	16 bits (high word)
11	CMP. L SrcOp,Rn	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the 32 bit form (.L) is used, but the source operand is a 16 bit constant, the upper 16 bits copied into the destination operand will be the upper 16 bits of R0.

DEC

Summary: Decrement
Operation: Register/Memory – 1 → Register
Assembler Syntax: DEC SrcOp

Description: Subtracts one from the source operand (no carry is subtracted or generated). Note that if the operand is a memory reference (**DEC [reg]** or **DEC [reg,imm16]**), then the destination is R0 and nothing is written back. In that case, you must write the value back into the source memory location by using ST; for example, the 6502 operation **DEC \$FF** can be done as **DEC.B [R0,\$ff]; ST.B R0,[R0,\$FF]** (see the notes at the end of the entry for ST for an explanation of this usage of the ST opcode). Because of the implicit write into R0, the **DEC** and associated **ST** execute atomically. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
–	✓	–	–	–	–	–	✓

Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	1	1	0	Addr Mode			1	1	Source Register			0	0	1	Data Width
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	DEC [reg]	Register Indirect
01	DEC [reg,offset16]	Register Indirect with Offset
10	DEC reg	Register

Source Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	DEC. B SrcOp	8 bits
01	DEC. W SrcOp	16 bits (low word)
10	DEC. H SrcOp	16 bits (high word)
11	DEC. L SrcOp	32 bits

Notes:

1. If the operand is R0 or a memory reference, interrupts will be disabled until the next instruction.

EOR

Summary: Bitwise Exclusive OR
Operation: Register/Memory ∇ Register \rightarrow Register
Assembler Syntax: EOR SrcOp, Rn

Description: Performs a bitwise “exclusive OR” of the source operand and the destination operand and stores the result in the destination register. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
-	✓	-	-	-	-	-	✓

Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	1	0	Addr Mode	1	1		Source Register			Destination Register			Data Width	
Immediate/offset LSB (only if used by addressing mode)								Immediate/offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	EOR [reg],reg	Register Indirect
01	EOR [reg,offset16],reg	Register Indirect with Offset
10	EOR reg,reg	Register
11	EOR imm16,reg	Immediate

Source/Destination Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon’s Register (general purpose)
101	R5	J	Jeri’s Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	EOR.B SrcOp,Rn	8 bits
01	EOR.W SrcOp,Rn EOR.W SrcOp,RnL	16 bits (low word)
10	EOR.H SrcOp,Rn EOR.W SrcOp,RnH	16 bits (high word)
11	EOR.L SrcOp,Rn	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the 32 bit form (.L) is used, but the source operand is a 16 bit constant, the upper 16 bits copied into the destination operand will be the upper 16 bits of R0.

INC

Summary: Increment
Operation: Register/Memory + 1 → Register
Assembler Syntax: INC SrcOp

Description: Adds one to the source operand (no carry is added or generated). Note that if the operand is a memory reference (`INC [reg]` or `INC [reg,imm16]`), then the destination is R0 and nothing is written back. In that case, you must write the value back into the source memory location by using ST; for example, the 6502 operation `INC $FF` can be done as `INC.B [R0,$ff]; ST.B R0,[R0,$FF]` (see the notes at the end of the entry for ST for an explanation of this usage of the ST opcode). Because of the implicit write into R0, the `INC` and associated `ST` execute atomically. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
-	✓	-	-	-	-	-	✓

Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	1	1	0	Addr Mode	1	1		Source Register	0	0	0			Data Width	
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	<code>INC [reg]</code>	Register Indirect
01	<code>INC [reg,offset16]</code>	Register Indirect with Offset
10	<code>INC reg</code>	Register

Source Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	INC. B SrcOp	8 bits
01	INC. W SrcOp	16 bits (low word)
10	INC. H SrcOp	16 bits (high word)
11	INC. L SrcOp	32 bits

Notes:

1. If the operand is R0 or a memory reference, interrupts will be disabled until the next instruction.

JSR

Summary: Jump to subroutine
Operation: $PC + (1 \text{ or } 3) \rightarrow [SP]$, $SP - (2 \text{ or } 4) \rightarrow SP$, Register/Imm16 $\rightarrow PC$
Assembler Syntax: JSR SrcOp

Description: Pushes the address of the next instruction, minus one, into the stack (this will be $PC + 1$ or $PC + 3$ depending on whether JSR instruction needs a 16 bit or 32 bit encoding), then jumps to the given address. The size of the operation may be specified as word or long. Word implies a jump to the lowest 64K. The strange return address semantics are to maintain backwards compatibility with the 6502.

Affected Flags:

C	Z	I	D	B	E	V	N
-	-	-	-	-	-	-	-

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	1	1	Addr Mode	1	1		Register	0	0	0	0	0	0	Width
Immediate/offset LSB (only if used by addressing mode)								Immediate/offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	JSR [reg]	Register Indirect
01	JSR [reg,offset16]	Register Indirect with Offset
10	JSR reg	Register
11	JSR imm16	Immediate

Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Width:

Encoding	Assembler Syntax	Width
0	JSR.W SrcOp	16 bits
1	JSR.L SrcOp	32 bits

Notes:

1. If the 32 bit form (.L) is used, but the operand is a 16 bit constant, the upper 16 bits copied into the program counter will be the upper 16 bits of R0.
2. The **RET** instruction used to return from the called subroutine **MUST** have the same operand width as the **JSR**!
3. When calling a subroutine with the 6502 3-byte **JSR**, you **must** return using **RTS**; when using the 65GZ032 **JSR.W** or **JSR.L**, you **must** return using the appropriate form of **RET**. **DO NOT MIX THESE UP!**

LCNTR

Summary: Load control register
Operation: Register/Memory → CNTR
Assembler Syntax: LCNTR SrcOp

Description: Loads the contents of a register or memory and stores it into the control (Flags) register.

Affected Flags:

C	Z	I	D	B	E	V	N	–	–	FDC	FIC	DCA	ICA	FTL	MMU
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

All flags are written to simultaneously.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	1	1	Addr Mode	1	1		Register	1	0	0	0			Width
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	LCNTR [reg]	Register Indirect
01	LCNTR [reg,offset 6]	Register Indirect with Offset
10	LCNTR [reg++]	Register Indirect with Autoincrement
11	LCNTR [--reg]	Register Indirect with Autodecrement

NOTE: The autoincrement/autodecrement addressing modes add or subtract the instruction's data width from the base register, i.e. opcode.L adds or subtracts 4 and opcode.W adds or subtracts 2.

Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Width:

Encoding	Assembler Syntax	Width
0	LCNTR.W SrcOp	16 bits
1	LCNTR.L SrcOp	32 bits

LD

Summary: Load from Memory
Operation: Memory → Register
Assembler Syntax: LD SrcOp, Rn

Description: Loads the source operand from memory and stores the result in the destination register. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
–	✓	–	–	–	–	–	✓

Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	0	0	Addr Mode	1	1		Source Register		Destination Register		Data Width			
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	LD [reg],reg	Register Indirect
01	LD [reg,offset16],reg	Register Indirect with Offset
10	LD [reg++],reg	Register Indirect with Autoincrement
11	LD [--reg],reg	Register Indirect with Autodecrement

NOTE: The autoincrement/autodecrement addressing modes add or subtract the instruction's data width from the base register, i.e. opcode.L adds or subtracts 4, opcode.W and opcode.H 2, and opcode.B 1.

Source/Destination Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	LD.B SrcOp,Rn	8 bits
01	LD.W SrcOp,Rn LD.W SrcOp,RnL	16 bits (low word)
10	LD.H SrcOp,Rn LD.W SrcOp,RnH	16 bits (high word)
11	LD.L SrcOp,Rn	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the autoincrement/autodecrement addressing modes are used and the destination register is SP (R6), the order is backwards: for autoincrement, the increment happens first, and for autodecrement, the decrement happens last. This is for backwards compatibility with the 6502's PUSH and POP opcodes.

LMMU

Summary: Load Memory Management Unit Register
Operation: Register/Memory → MMU Register
Assembler Syntax: LMMU SrcOp,Mn

Description: This instruction loads the value of MMU control register 0 or 1 from the given memory location.

Affected Flags:

C	Z	I	D	B	E	V	N
–	✓	–	–	–	–	–	✓

Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	1	1	Addr Mode	1	1		Register	1	1	0	0			MMU Reg
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	LMMU [reg]	Register Indirect
01	LMMU [reg,offset16]	Register Indirect with Offset
10	LMMU [reg++]	Register Indirect with Autoincrement
11	LMMU [--reg]	Register Indirect with Autodecrement

NOTE: As far as the autoincrement/autodecrement addressing modes are concerned, LMMU operates on 32 bit data, so the amount added or subtracted from the base register is always 4.

Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

MMU Register:

Encoding	Register	Description
0	M0	Unknown
1	M1	Unknown

MIRROR

Summary: Do a mirror image of the bits in a byte/word

Operation:

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

 →

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Assembler Syntax: MIRROR SrcOp

Description: This does a mirror image of all the bits in a byte, or those in a 16 bit word. Note that if the operand is a memory reference (MIRROR [reg] or MIRROR [reg,imm16]), then the destination is R0 and nothing is written back. In that case, you must write the value back into the source memory location by using ST. Because of the implicit write into R0, the MIRROR and associated ST execute atomically. The size of the operation may be specified as byte or word (low only).

Affected Flags:

C	Z	I	D	B	E	V	N
-	✓	-	-	-	-	-	✓

Z — Set if the result is zero; cleared otherwise.

N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	1	1	0	Addr Mode	1	1		Source Register	0	1	1			Data Width	
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	MIRROR [reg]	Register Indirect
01	MIRROR [reg,offset16]	Register Indirect with Offset
10	MIRROR reg	Register

Source Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	MIRROR. B SrcOp	8 bits
01	MIRROR. W SrcOp	16 bits (low word)

Notes:

1. If the operand is R0 or a memory reference, interrupts will be disabled until the next instruction.

MOV

Summary: Move register or immediate to register
Operation: Register/Imm16 → Register
Assembler Syntax: MOV SrcOp, Rn

Description: Moves the contents of one register (or a 16-bit constant) into another register. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
–	✓	–	–	–	–	–	✓

Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	1	0	0	Addr Mode	1	1	Source Register			Destination Register			Data Width	
Immediate LSB (only if used by addressing mode)								Immediate MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
0	MOV reg,reg	Register
1	MOV imm16,reg	Immediate

Source/Destination Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon’s Register (general purpose)
101	R5	J	Jeri’s Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	MOV.B SrcOp,Rn	8 bits
01	MOV.W SrcOp,Rn MOV.W SrcOp,RnL	16 bits (low word)
10	MOV.H SrcOp,Rn MOV.W SrcOp,RnH	16 bits (high word)
11	MOV.L SrcOp,Rn	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. To load 8-bit constants into registers R1, R2 or R3 you might want to consider LDA, LDX and LDY, since these use only 2 bytes of code space instead of the 4 that MOV needs.

MOVX

Summary: Move register or immediate to register, with sign extend
Operation: Register/Imm16 → Register (sign extended)
Assembler Syntax: MOVX SrcOp, Rn

Description: Moves the contents of one register (or a 16-bit constant) into another register. Sign extension will also be done, i.e. the most significant bit of the source operand will be replicated to all bits to the left of it; this has the effect (in the 2's complement representation) of storing an integer into a wider word while preserving its sign and magnitude. The size of the operation may be specified as byte, word (high or low) or long; note that this width setting applies only to the source operand – the destination operand will always be long (32 bits).

Affected Flags:

C	Z	I	D	B	E	V	N
–	✓	–	–	–	–	–	✓

Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	1	0	1	Addr Mode	1	1	Source Register			Destination Register			Data Width	
Immediate LSB (only if used by addressing mode)								Immediate MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
0	MOVX reg,reg	Register
1	MOVX imm16,reg	Immediate

Source/Destination Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	MOVX. B SrcOp,Rn	8 bits
01	MOVX. W SrcOp,Rn MOVX. W SrcOp,Rn L	16 bits (low word)
10	MOVX. H SrcOp,Rn MOVX. W SrcOp,Rn H	16 bits (high word)
11	MOVX. L SrcOp,Rn	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. The entire 32 bits of the destination register will be overwritten; data width applies only to the source.

OR

Summary: Bitwise OR
Operation: Register/Memory \vee Register \rightarrow Register
Assembler Syntax: OR SrcOp, Rn

Description: Performs a bitwise “OR” of the source operand and the destination operand and stores the result in the destination register. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
–	✓	–	–	–	–	–	✓

Z — Set if the result is zero; cleared otherwise.
N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	Addr Mode	1	1		Source Register	Destination Register			Data Width			
Immediate/offset LSB (only if used by addressing mode)								Immediate/offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	OR [reg],reg	Register Indirect
01	OR [reg,offset16],reg	Register Indirect with Offset
10	OR reg,reg	Register
11	OR imm16,reg	Immediate

Source/Destination Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon’s Register (general purpose)
101	R5	J	Jeri’s Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	OR. B SrcOp,Rn	8 bits
01	OR. W SrcOp,Rn OR. W SrcOp,Rn L	16 bits (low word)
10	OR. H SrcOp,Rn OR. W SrcOp,Rn H	16 bits (high word)
11	OR. L SrcOp,Rn	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the 32 bit form (.L) is used, but the source operand is a 16 bit constant, the upper 16 bits copied into the destination operand will be the upper 16 bits of R0.

RET

Summary: Return from subroutine
Operation: $SP + (2 \text{ or } 4) \rightarrow SP, [SP] \rightarrow PC, PC + 1 \rightarrow PC$
Assembler Syntax: RET

Description: Pops a return address from the stack and jumps to that address, plus one. The size of the operation may be specified as word or long, but must match the corresponding JSR. The strange return address semantics are to maintain backwards compatibility with the 6502.

Affected Flags:

C	Z	I	D	B	E	V	N
-	-	-	-	-	-	-	-

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Width
1	0	1	1	-	-	1	1	-	-	-	0	0	1	0		

Width:

Encoding	Assembler Syntax	Width
0	RET.W	16 bits
1	RET.L	32 bits

Notes:

1. The **RET** instruction used to return from the called subroutine **MUST** have the same operand width as the **JSR**!
2. When calling a subroutine with the 6502 3-byte **JSR**, you **must** return using **RTS**; when using the 65GZ032 **JSR.W** or **JSR.L**, you **must** return using the appropriate form of **RET**. **DO NOT MIX THESE UP!**

RETI

Summary: Return from interrupt
Operation: $SP + 2 \rightarrow SP$, $[SP] \rightarrow \text{Flags}$, $SP + 2 \rightarrow SP$, $[SP] \rightarrow PC$
Assembler Syntax: RETI

Description: Pops the flags register and a return address from the stack and jumps to that address. The size of the operation may be specified as word or long, but must match the corresponding TRAP.
(what about hardware interrupts? Are they going to be 16 or 32 bit vectors?)

Affected Flags:

C	Z	I	D	B	E	V	N
✓	✓	✓	✓	✓	✓	✓	✓

All flags are retrieved from the stack.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
1	0	1	1	-	-	1	1	-	-	-	0	1	1	-		Width

Width:

Encoding	Assembler Syntax	Width
0	RETI.W	16 bits
1	RETI.L	32 bits

(this page intentionally left blank)

SCNTR

Summary: Store control (Flags) register
Operation: CNTR → Register/Memory
Assembler Syntax: SCNTR SrcOp

Description: Loads the control (Flags) register and stores it into the given register.

Affected Flags:

C	Z	I	D	B	E	V	N	-	-	FDC	FIC	DCA	ICA	FTL	MMU
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	1	1	Addr Mode	1	1		Register	1	0	0	1		Width	
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	SCNTR [reg]	Register Indirect
01	SCNTR [reg,offset16]	Register Indirect with Offset
10	SCNTR [reg++]	Register Indirect with Autoincrement
11	SCNTR [--reg]	Register Indirect with Autodecrement

NOTE: The autoincrement/autodecrement addressing modes add or subtract the instruction's data width from the base register, i.e. opcode.L adds or subtracts 4 and opcode.W adds or subtracts 2.

Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Width:

Encoding	Assembler Syntax	Width
0	SCNTR.W SrcOp	16 bits
1	SCNTR.L SrcOp	32 bits

SFLAG

Summary: Set one bit in the Flags register
Operation: 1 → Flags[k]
Assembler Syntax: SFLAG imm4

Description: Sets any bit in the Flags register to '1'.

Affected Flags:

C	Z	I	D	B	E	V	N	–	–	FDC	FIC	DCA	ICA	FTL	MMU
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

You can set any flag bit.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	1	1	1	Flag (bit 0)	1	1	Flag (bits 3–1)			1	0	1	–	–

Flag Number:

Encoding	Abbrev	Name	Encoding	Abbrev	Name
0000	C	Carry	1000	–	–
0001	Z	Zero	1001	–	–
0010	I	Interrupt Disable	1010	FDC	Flush Data Cache
0011	D	Decimal	1011	FIC	Flush Inst. Cache
0100	B	Break	1100	DCA	Data Cache Enabled
0101	E	(Emulation?)	1101	ICA	Inst. Cache Enabled
0110	V	oVerflow	1110	FTL	Flush Translation Lookup
0111	N	Negative	1111	MMU	MMU Enabled

(this page intentionally left blank)

SHL

Summary: Shift left
Operation: $C \leftarrow$

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

 $\leftarrow 0 (0,0,0)$
Assembler Syntax: SHL SrcOp,<shift amount>

Description: Shifts all the bits of the source operand left by the given amount, which is a constant between 1 and 4. Zeroes will be shifted in from the right, and the carry flag will contain the last bit that got shifted out. All other shifted out bits will be lost. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
✓	✓	–	–	–	–	–	✓

C — Set if a carry is generated; cleared otherwise.
 Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	1	0	0	Addr Mode	1	1		Source Register	0	Shift Amount		Data Width			
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	SHL [reg],cnt	Register Indirect
01	SHL [reg,offset 1 6],cnt	Register Indirect with Offset
10	SHL reg,cnt	Register

Source Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Shift Amount:

Encoding	Bits to Shift
00	4
01	1
10	2
11	3

Data Width:

Encoding	Assembler Syntax	Width
00	SHL.B SrcOp,cnt	8 bits
01	SHL.W SrcOp,cnt	16 bits (low word)
10	SHL.H SrcOp,cnt	16 bits (high word)
11	SHL.L SrcOp,cnt	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the destination is a memory address, the intermediate register (R0/Ri) will also contain the result.
3. You can use this to multiply integers by powers of two: SHL.L R1,k is equivalent to $R1 \times 2^k$

SHLC

Summary: Shift left with carry
Operation: $C \leftarrow \boxed{7 \ 6 \ 5 \ 4 \ 3 \ 2 \ 1 \ 0} \leftarrow C \ (C, C, C)$
Assembler Syntax: SHL SrcOp, <shift amount>

Description: Shifts all the bits of the source operand left by the given amount, which is a constant between 1 and 4. The value of the carry flag will be shifted in from the right, and the carry flag will contain the last bit that got shifted out. All other shifted out bits will be lost. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
✓	✓	–	–	–	–	–	✓

C — Set if a carry is generated; cleared otherwise.
 Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	1	0	0	Addr Mode	1	1		Source Register	1	Shift Amount			Data Width		
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	SHLC [reg],cnt	Register Indirect
01	SHLC [reg,offset16],cnt	Register Indirect with Offset
10	SHLC reg,cnt	Register

Source Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Shift Amount:

Encoding	Bits to Shift
00	4
01	1
10	2
11	3

Data Width:

Encoding	Assembler Syntax	Width
00	SHLC. B SrcOp,cnt	8 bits
01	SHLC. W SrcOp,cnt	16 bits (low word)
10	SHLC. H SrcOp,cnt	16 bits (high word)
11	SHLC. L SrcOp,cnt	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the destination is a memory address, the intermediate register (R0/Ri) will also contain the result.

SHR

Summary: Shift right
Operation: (0,0,0) 0 →

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

 → C
Assembler Syntax: SHR SrcOp,<shift amount>

Description: Shifts all the bits of the source operand right by the given amount, which is a constant between 1 and 4. Zeroes will be shifted in from the left, and the carry flag will contain the last bit that got shifted out. All other shifted out bits will be lost. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
✓	✓	–	–	–	–	–	✓

C — Set if a carry is generated; cleared otherwise.
 Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	1	0	1	Addr Mode	1	1		Source Register	0	Shift Amount				Data Width	
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	SHR [reg],cnt	Register Indirect
01	SHR [reg,offset16],cnt	Register Indirect with Offset
10	SHR reg,cnt	Register

Source Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Shift Amount:

Encoding	Bits to Shift
00	4
01	1
10	2
11	3

Data Width:

Encoding	Assembler Syntax	Width
00	SHR.B SrcOp,cnt	8 bits
01	SHR.W SrcOp,cnt	16 bits (low word)
10	SHR.H SrcOp,cnt	16 bits (high word)
11	SHR.L SrcOp,cnt	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the destination is a memory address, the intermediate register (R0/Ri) will also contain the result.
3. You can use this to divide unsigned integers by powers of two: **SHR.L R1,k** is equivalent to $R1 \div (2^k)$

SHRC

Summary: Shift right with carry
Operation: (C,C,C) C →

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

 → C
Assembler Syntax: SHR SrcOp,<shift amount>

Description: Shifts all the bits of the source operand right by the given amount, which is a constant between 1 and 4. The value of the carry flag will be shifted in from the left, and the carry flag will contain the last bit that got shifted out. All other shifted out bits will be lost. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
✓	✓	–	–	–	–	–	✓

C — Set if a carry is generated; cleared otherwise.
 Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	1	0	1	Addr Mode	1	1		Source Register	1	Shift Amount		Shift Amount		Data Width	
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	SHRC [reg],cnt	Register Indirect
01	SHRC [reg,offset 16],cnt	Register Indirect with Offset
10	SHRC reg,cnt	Register

Source Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Shift Amount:

Encoding	Bits to Shift
00	4
01	1
10	2
11	3

Data Width:

Encoding	Assembler Syntax	Width
00	SHRC.B SrcOp,cnt	8 bits
01	SHRC.W SrcOp,cnt	16 bits (low word)
10	SHRC.H SrcOp,cnt	16 bits (high word)
11	SHRC.L SrcOp,cnt	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the destination is a memory address, the intermediate register (R0/Ri) will also contain the result.
3. You can use this to divide signed integers by powers of two: SHRC.L R1,k is equivalent to $R1 \div (2^k)$

SMMU

Summary: Store Memory Management Unit Register
Operation: MMU Register → Register/Memory
Assembler Syntax: SMMU Mn,SrcOp

Description: This instruction stores the value of MMU control register 0 or 1 into the given memory location.

Affected Flags:

C	Z	I	D	B	E	V	N
–	✓	–	–	–	–	–	✓

Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	1	1	Addr Mode	1	1		Register	1	1	0	1		MMU Reg	
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	SMMU [reg]	Register Indirect
01	SMMU [reg,offset16]	Register Indirect with Offset
10	SMMU [reg++]	Register Indirect with Autoincrement
11	SMMU [--reg]	Register Indirect with Autodecrement

NOTE: As far as the autoincrement/autodecrement addressing modes are concerned, SMMU operates on 32 bit data, so the amount added or subtracted from the base register is always 4.

Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon’s Register (general purpose)
101	R5	J	Jeri’s Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

MMU Register:

Encoding	Register	Description
0	M0	Unknown
1	M1	Unknown

ST

Summary: Load from Memory
Operation: Register → Memory
Assembler Syntax: ST Rn, SrcOp

Description: Loads the source operand from memory and stores the result in the destination register. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
–	✓	–	–	–	–	–	✓

Z — Set if the result is zero; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	0	1	Addr Mode	1	1		Destination Register		Source Register			Data Width		
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	ST reg,[reg]	Register Indirect
01	ST reg,[reg,offset] 6	Register Indirect with Offset
10	ST reg,[reg++]	Register Indirect with Autoincrement
11	ST reg,[--reg]	Register Indirect with Autodecrement

NOTE: The autoincrement/autodecrement addressing modes add or subtract the instruction's data width from the base register, i.e. opcode.L adds or subtracts 4, opcode.W and opcode.H 2, and opcode.B 1.

Source/Destination Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	ST.B Rn,SrcOp	8 bits
01	ST.W Rn,SrcOp ST.W RnL,SrcOp	16 bits (low word)
10	ST.H Rn,SrcOp ST.W RnH,SrcOp	16 bits (high word)
11	ST.L Rn,SrcOp	32 bits

Notes:

1. If the autoincrement/autodecrement addressing modes are used and the destination register is SP (R6), the order is backwards: for autoincrement, the increment happens first, and for autodecrement, the decrement happens last. This is for backwards compatibility with the 6502's PUSH and POP opcodes.
2. When using R0 both for address and data, (eg ST.L R0,[R0 + imm16]), remember that data has priority over address. So when R0 is needed for both address and data, the address reference is zero and the data reference is used. This means that ST R0,[R0+\$nnnn] does a store of the temp value into the offset from zero (R0). This is useful when using ST in conjunction with INC or DEC.

SUB

Summary: Binary Subtraction (no carry)
Operation: Register/Memory – Register → Register
Assembler Syntax: SUB SrcOp, Rn

Description: Subtracts the destination from the source operand and stores the result in the destination register. The size of the operation may be specified as byte, word (high or low) or long. Note that this ignores the carry bit (see SUBC).

Affected Flags:

C	Z	I	D	B	E	V	N
✓	✓	–	–	–	–	✓	✓

C — Set if a carry is generated; cleared otherwise.
 Z — Set if the result is zero; cleared otherwise.
 V — Set if an overflow is generated; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	1	0	1	Addr Mode	1	1		Source Register		Destination Register		Data Width			
Immediate/offset LSB (only if used by addressing mode)								Immediate/offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	SUB [reg],reg	Register Indirect
01	SUB [reg,offset16],reg	Register Indirect with Offset
10	SUB reg,reg	Register
11	SUB imm16,reg	Immediate

Source/Destination Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	SUB.B SrcOp,Rn	8 bits
01	SUB.W SrcOp,Rn SUB.W SrcOp,RnL	16 bits (low word)
10	SUB.H SrcOp,Rn SUB.W SrcOp,RnH	16 bits (high word)
11	SUB.L SrcOp,Rn	32 bits

Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the 32 bit form (.L) is used, but the source operand is a 16 bit constant, the upper 16 bits copied into the destination operand will be the upper 16 bits of R0.

SUBC

Summary: Binary Subtraction (with carry)
Operation: Register/Memory – Register – Carry → Register
Assembler Syntax: SUBC SrcOp, Rn

Description: Subtracts the destination from the source operands, including the carry bit of the Flags register, and stores the result in the destination register. The size of the operation may be specified as byte, word (high or low) or long.

Affected Flags:

C	Z	I	D	B	E	V	N
✓	✓	–	–	–	–	✓	✓

C — Set if a carry is generated; cleared otherwise.
 Z — Set if the result is zero; cleared otherwise.
 V — Set if an overflow is generated; cleared otherwise.
 N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	1	1	0	Addr Mode	1	1		Source Register		Destination Register				Data Width	
Immediate/offset LSB (only if used by addressing mode)								Immediate/offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	SUBC [reg],reg	Register Indirect
01	SUBC [reg,offset16],reg	Register Indirect with Offset
10	SUBC reg,reg	Register
11	SUBC imm16,reg	Immediate

Source/Destination Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon's Register (general purpose)
101	R5	J	Jeri's Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
00	SUBC.B SrcOp,Rn	8 bits
01	SUBC.W SrcOp,Rn SUBC.W SrcOp,RnL	16 bits (low word)
10	SUBC.H SrcOp,Rn SUBC.W SrcOp,RnH	16 bits (high word)
11	SUBC.L SrcOp,Rn	32 bits

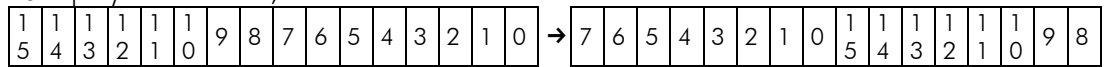
Notes:

1. If the destination is R0, interrupts will be disabled until the next instruction.
2. If the 32 bit form (.L) is used, but the source operand is a 16 bit constant, the upper 16 bits copied into the destination operand will be the upper 16 bits of R0.

SWAP

Summary: Swap bytes in a word, or words in a dword

Operation:



Assembler Syntax: SWAP SrcOp

Description: Swaps the two bytes in a 16 bit word, or the two 16 bit words in a 32 bit dword. Note that if the operand is a memory reference (**SWAP [reg]** or **SWAP [reg,imm16]**), then the destination is R0 and nothing is written back. In that case, you must write the value back into the source memory location by using ST. Because of the implicit write into R0, the **SWAP** and associated **ST** execute atomically. The size of the operation may be specified as word (high or low) or long. Please note the meaning is slightly different, see the “Data Width” table for details.

Affected Flags:

C	Z	I	D	B	E	V	N
–	✓	–	–	–	–	–	✓

Z — Set if the result is zero; cleared otherwise.

N — Set if the result is negative; cleared otherwise.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	1	1	0	Addr Mode	1	1		Source Register	0	1	0			Data Width	
Offset LSB (only if used by addressing mode)								Offset MSB (only if used by addressing mode)							

Addressing modes:

Encoding	Assembler Syntax	Addressing mode
00	SWAP [reg]	Register Indirect
01	SWAP [reg,offset16]	Register Indirect with Offset
10	SWAP reg	Register

Source Register:

Encoding	Register	Alt. Name	Description
000	R0	Z or Ri	Zero, or Implicit Register
001	R1	A	Accumulator, also general purpose register
010	R2	X	X offset register, also general purpose register
011	R3	Y	Y offset register, also general purpose register
100	R4	G	Gideon’s Register (general purpose)
101	R5	J	Jeri’s Register (general purpose)
110	R6	SP	Stack Pointer
111	R7	PC	Program Counter

Data Width:

Encoding	Assembler Syntax	Width
01	SWAP.W SrcOp	Swap bytes in low 16 bit word
10	SWAP.H SrcOp	Swap bytes in high 16 bit word
11	SWAP.L SrcOp	Swap high and low 16 words

Notes:

1. If the operand is R0 or a memory reference, interrupts will be disabled until the next instruction.

TRAP

Summary: Trap to OS
Operation: PC + 2 → [SP], SP - 2 → SP, Flags → [SP], SP - 2 → SP, Memory → PC
Assembler Syntax: TRAP imm5

Description: Pushes the address of the next instruction, as well as the Flags register into the stack, then jumps to an address specified in a table in memory ([details not yet specified](#)). The size of the operation may be specified as word or long. Word implies that the table contains 16-bit entries (all pointing to routines in the lowest 64K), otherwise the table contains 32 bit addresses.

Affected Flags:

C	Z	I	D	B	E	V	N
-	-	✓	-	✓	-	-	-

I — Always set.
 B — Always cleared.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	1	1	Trap number (bits 1–0)		1	1	Trap number (bits 4–2)		0	1	0	0	Width	

Trap Number:

Encoding	Name	Description	Encoding	Name	Description
00000	Unknown trap		10000	Unknown trap	
00001	Unknown trap		10001	Unknown trap	
00010	Unknown trap		10010	Unknown trap	
00011	Unknown trap		10011	Unknown trap	
00100	Unknown trap		10100	Unknown trap	
00101	Unknown trap		10101	Unknown trap	
00110	Unknown trap		10110	Unknown trap	
00111	Unknown trap		10111	Unknown trap	
01000	Unknown trap		11000	Unknown trap	
01001	Unknown trap		11001	Unknown trap	
01010	Unknown trap		11010	Unknown trap	
01011	Unknown trap		11011	Unknown trap	
01100	Unknown trap		11100	Unknown trap	
01101	Unknown trap		11101	Unknown trap	
01110	Unknown trap		11110	Unknown trap	
01111	Unknown trap		11111	Unknown trap	

Width:

Encoding	Assembler Syntax	Width
0	TRAP.W num	16 bits
1	TRAP.L num	32 bits

TRAPB

Summary: Trap to OS, set the B bit
Operation: PC + 2 → [SP], SP - 2 → SP, Flags → [SP], SP - 2 → SP, Memory → PC
Assembler Syntax: TRAPB imm5

Description: Pushes the address of the next instruction, as well as the Flags register into the stack, then jumps to an address specified in a table in memory ([details not yet specified](#)). The size of the operation may be specified as word or long. Word implies that the table contains 16-bit entries (all pointing to routines in the lowest 64K), otherwise the table contains 32 bit addresses.

Affected Flags:

C	Z	I	D	B	E	V	N
-	-	✓	-	✓	-	-	-

I — Always set.
 B — Always set.

Instruction Format:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	1	1	Trap number (bits 1-0)		1	1	Trap number (bits 4-2)		0	1	0	1	Width	

Trap Number:

Encoding	Name	Description	Encoding	Name	Description
00000	Unknown trap		10000	Unknown trap	
00001	Unknown trap		10001	Unknown trap	
00010	Unknown trap		10010	Unknown trap	
00011	Unknown trap		10011	Unknown trap	
00100	Unknown trap		10100	Unknown trap	
00101	Unknown trap		10101	Unknown trap	
00110	Unknown trap		10110	Unknown trap	
00111	Unknown trap		10111	Unknown trap	
01000	Unknown trap		11000	Unknown trap	
01001	Unknown trap		11001	Unknown trap	
01010	Unknown trap		11010	Unknown trap	
01011	Unknown trap		11011	Unknown trap	
01100	Unknown trap		11100	Unknown trap	
01101	Unknown trap		11101	Unknown trap	
01110	Unknown trap		11110	Unknown trap	
01111	Unknown trap		11111	Unknown trap	

Width:

Encoding	Assembler Syntax	Width
0	TRAPB.W num	16 bits
1	TRAPB.L num	32 bits