#### VGA Controller

#### VGA\$STATE Bits

Bit	Description
11	Enable R/W offset register if set.
10	Enable display offset register if set.
9	Busy (wait for 0 before issuing command).
8	Clear screen (set until completion).
7	Enable VGA controller.
6	Enable hardware cursor.
5	Enable hardware cursor blinking.
4	Hardware cursor mode:
	Small if set, large if cleared.
20	Display color (RGB).

VGA\$CR\_X X coordinate of next char to be displayed.

VGA\$CR\_Y Y coordinate of next char to be displayed.

VGA\$CHAR Writing a byte to this register causes it to be displayed on the current X/Y coordinate on the screen. Reading from this register yields the character at the current display coordinate.

VGA\$OFFS\_DISPLAY This register holds the offset in bytes that is to be used when displaying the video RAM. To scroll one line forward, simply add 0x0050 to this register. For this to work, bit 10 in VGA\$STATE has to be set.

 $VGA\$OFFS_RW$  Similar to  $VGA\$OFFS_DISPLAY$  – controls the offset for read/write accesses to the display memory.

### **USB-Keyboard**

#### IO\$KBD\_STATE

Bit	Description
0	Set if an unread character is available.
1	Key pressed (val in bits 158
$2\dots 4$	Keyboard layout:
	000: US keyboard
	001: German keyboard
57	Key modifier bit mask:
	5: shift, 6: alt, 7: ctrl

### Cycle Counter

#### CYC\$STATE

Bit	Description
0	Reset counter and start counting.
1	1: count, 0: inhibit

#### EAE

#### IO\$EAE\_CSR

Bit	Description
,	Operation (MULU, MULS, DIVU, DIVS) Busy if set

#### UART

SUBR:

#### IO\$UART\_SRA

Bit	Description
0	Character received.
1	Transmitter ready for next character.

## **Code Examples**

### Typical Subroutine Call

MOVE ..., R8 ; Setup parameters ...

RSUB SUBR, 1 ; Call subroutine ...

INCRB ; Get free reg. set ...

DECRB ; Restore reg. set MOVE @R13++, R15 ; RET

## Compute $\sum_{i=0}^{16} 0 \times 0010$

.ORG 0x8000

XOR RO, RO; Clear RO
MOVE 0x0010, R1; Upper limit
LOOP: ADD R1, RO; One summation
SUB 0x0001, R1; Decrement i
ABRA LOOP, !Z; Loop if not zero

# QNICE programming card

## ISA v1.7

### October 18, 2020

#### General

QNICE features 16 bit words, 16 registers, 4 addressing modes, and a 16 bit address space (16 bit words, upper 256 words reserved for memory mapped I/O).

## Registers

All in all there are 16 general purpose registers ( GPRs ) available:

RO		R7	R8		R13	R14	R15
----	--	----	----	--	-----	-----	-----

R0...R7: GPRs, actually these are a window into a register bank holding  $256 \times 8$  such registers.

R13: Stack pointer (SP).

R14: Statusregister (SR).

R15: Program counter (PC).

## Statusregister

rbank — —	VN	ZC	X	1
-----------	----	----	---	---

1: Always set to 1.

X: Used by shift instructions only.

C: Carry flag.

Z: 1 if the last result was 0x0000.

N: 1 if the last result was negative.

V: 1 if the last operation caused an overflow, i.e. two positive operands yielded a negative result or vice versa.

CPU saves the contents of R8 to R15 in eight shadow registers which can be accessed with the RTI-instruction. An ISR must be left with the RTI-instruction. Interrupts can not be nested.

# tuquO/tuquI

 $\rm I/O$  devices are memory mapped, their respective control and data registers occupy the topmost 256 words of memory.

wor bilsv .xsm IMGH	0xFF36	VGA\$HDMI_V_MAX
HDMI max. valid col.	0xFF36	VGA\$HDMI_H_MAX
HDMI min. valid col.	0xFF36	VGA\$HDMI_H_MIN
B/W RAM offset	0xFF35	AGV#OŁŁZ-KM
Display RAM offset	0xFF34	AGV#OLEZ-DIZBFVA
Character code	0xFF33	VGA\$CHAR
Cursor y-position	0xFF32	VGA\$CR_Y
Cursor X-position	0xFF31	VGA\$CR_X
VGA status register	0xFF30	AGA\$STATE
Timer I ISR addr.	0xFF28	IO\$TIMER_1_INT
Timer 1 counter	0xFF28	IO\$TIMER_1_CUT
Timer 1 prescaler	0xFF28	IO\$TIMER_1_PRE
Timer 0 ISR addr.	0xFF28	IO\$TIMER_O_INT
Timer 0 counter	0xFF28	IO\$TIMER_O_CUT
Timer 0 prescaler	0xFF28	IO\$LIWEE-O-PRE
Command and status reg.	0xFF25	IO\$SD~CSK
Error code	0×FF24	IO\$SD_ERROR
Byte in 512 byte bfr.	0xFF23	ATAC_CI\$01
Ptr. to 512 byte bfr.	0xFF22	IO\$SD_ADDR_POS
SD card high addr.	0xFF21	IH_ADDA_d2\$OI
SD card low addr.	0xFF20	IO\$SD_ADDR_LO
EAE command & status reg.	0xFF1C	IO\$EVE-CSR
HAE high result	0xFF1B	IO\$EAE_RESULT_HI
EAE low result	OxFF1A	IO\$EAE_RESULT_LO
EAE 2nd operand	0xFF19	IO\$EFE-OPERAND_1
EAE 1st operand	0xFF18	IO\$EVE-OPERAND-O
UART receive register	0xFF13	AAHT_TAAU\$OI
UART receive register	0xFF12	IO\$UART_TARRA
TAAU status register	0xFF11	AA2_TAAU\$OI
TAAU status register	0xFF10	x1AM_TAAU\$OI
Cycle counter status	0xFF0F	IO\$INS_STATE
Cycle counter high	OXFF0E	IH-SNI\$OI
Cycle counter middle	OxFFOD	QIM-SNI\$OI
Cycle counter low	0×FF0C	OT-SNI\$OI
Cycle counter status	0×FF0B	IO\$CYC_STATE
Cycle counter high	A077x0	IO\$CAC~HI
Cycle counter middle	0xFF09	IO\$CAC~WID
Cycle counter low	0xFF08	IO\$CAC_LO
USB-keyboard data	0xFF04	IO\$KBD_DATA
USB-keyboard state	0xFF03	IO\$KBD-STATE
Mask register	0xFF02	IO\$TIL_MASK
TIL-display	0xFF01	IO\$TIL_DISPLAY
Switch register	0xFF00	IO\$SMITCH_REG
Start of I/O area	0xFF00	IO\$BYSE
Description		
Toitairasa(I	ssərbbA	Label

### CMB

The CMP (compare) instruction can be used for signed as well as for unsigned comparisons:

Ţ	0	Ţ	0	src>dst	
0	Ţ	0	r   tsb=ore		
0	0	0	0	src <dst< td=""></dst<>	
Λ	Z	N	Z		
bəngis bəngisnu			un		
Flags				Condition	

## Addressing Modes

operand		
memory cell addressed by Rxx as		
Decrement Rxx and then use the	XXA0	11
then increment Rxx		
the contents of Rxx as operand and		
Use the memory cell addressed by	#+xxAØ	10
the contents of Rxx as operand		
Use the memory cell addressed by	gRxx	10
Use Rxx as operand	яхх	00
Description	Notation	stid sboM

#### Shortcuts

The file sysdef.asm (part of the monitor) defines some shortcuts which facilitate write- and readability of QNICE assembler code:

RIS	ЪС
R14	<b>ม</b> ร
R13	ЗЪ
γ ,x AUSA	SASCALL(x, y)
ABRA R15, 1	NOP
WONE GB13++' B12	RET
Implementation	Sportcut

## Interrupts

In case of a hardware interrupt, the processor expects the address of the interrupt service routine (ISR) on the data bus. In case of a software interrupt (IMT-instruction) the ISR address is specified by the data part of the instruction. When an interrupt occurs, the

The upper eight bits of  $\mathtt{SR}$  hold the pointer to the register window. Changing the value stored here will yield a different set of GPRs  $\mathtt{RO...R7}$  which is especially useful for subroutine calls (a stack of registers, so to speak).

## Instruction Set

QNICE features 14 basic instructions, four jump/branch instructions, and four adressing modes.

Relative subroutine call	dst, [!] cond	RSUB	F
Relative branch	dat, [!] cond	ARBA	F
Absolut subroutine call	dat, [!] cond	ASUS	F
Absolute branch	dat, [!] cond	ABBA	F
Exchange shadow register	tab ,tamoo	EXC	E
Decrement register bank address		DECKB	E
Increment register bank address		INCEB	E
Issue software interrupt	tsb	TNI	E
Return from interrupt		ITA	E
Halt the processor		TJAH	E
		reserved	D
compare arc with dat	tab , ara	CWP	ລ
dst := dst ^ src	src, dat	яох	В
dst := dst   src	tab , ara	Я0	A
dst := dst & src	arc, dat	AND	6
dst := !src	arc, dat	TON	8
((src >> 8) & OxFF)			
dst := ((src << 8) & 0xFF00)	tab , ara	<b>GAWS</b>	7
X of flik C, shift to X	arc, dat	SHR	9
D of thirds ,X driw llft , sit to C	arc, dat	THS	9
dst := dst - src - C	tab , ara	SUBC	₽
dst := dst - src	arc, dat	SUB	ε
dst := dst + src + C	arc, dat	ADDC	2
dst := dst + src	arc, dat	ADD	Ţ
dst := src	arc, dat	WOVE	0
Effect	Operands	ıısul	Opc

### Basic Instructions (opcodes 0...C)

ebom tab	xxr tab	arc mode	src rxx	obcoge
tid 2	tid ₽	tid 2	tid ₽	tid ₽

#### Control instructions

ebom tab	xxr tsb	command	obcoqe=E
tid 2	tid ₽	tid 8	tid ₽

### Jumps and Branches

conditio	condition	тоде	src mode	SIC IXX	obcoqe=E
select	negate				
3 bit	id 1	2 bit	2 bit	tid ₽	tid ₽