`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	Function/cursor/... key pressed. The value is stored in bits `15...8`.
2...4	Keyboard layout: `000`: US keyboard `001`: German keyboard
5...7	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

UART

`IO$UART_SRA`

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

Code Examples

Typical Subroutine Call

```
        MOVE ..., R8     ; Setup subroutine
                         ; parameters
        ...
        RSUB SUBR, 1     ; Call subroutine
        ...
SUBR:   ADD 0x0100, R14  ; Get free lower
                         ; register set
        ...
        SUB 0x0100, R14  ; Restore lower
                         ; register bank
        MOVE @R13++, R15 ; RET
```

Compute $\sum_{i=0}^{16}$ 0x0010

```
        .ORG 0x8000
        XOR R0,      R0 ; Clear R0
        MOVE 0x0010, R1 ; Upper limit
LOOP:   ADD R1,      R0 ; One summation
        SUB 0x0001,  R1 ; Decrement i
        ABRA LOOP,   !Z ; Loop if not zero
        HALT
```

QNICE programming card

May 4, 2016

General

QNICE is a 16 bit processor featuring four addressing modes, 16 registers and a 16 bit address space of 16 bit words, the upper 1 kW page is reserved for memory mapped I/O.

Registers

All in all there are 16 general purpose registers (*GPR*s) available:

R0	...	R7	R8	...	R13	R14	R15

`R0...R7:` General purpose registers, actually these are a window into a register bank holding 256×8 such registers.

`R13:` Stack pointer (`SP`).

`R14:` Statusregister (`SR`).

`R15:` Program counter (`PC`).

`SR`

rbank	M	I	V	N	Z	C	X	1

`1:` Always set to 1

X: 1 if the last result was 0xFFFF

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: 1 if an interrupt occured

M: If set to 1, maskable interrupts are allowed

The upper eight bits of SR hold the pointer to the register window. Changing the value stored here will yield a different set of GPRs R0...R7 which is especially useful for subroutine calls.

Instruction Set

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

Basic Instructions

4 bit	4 bit	2 bit	4 bit	2 bit
opcode	src rxx	src mode	dst rxx	dst mode

Opc	Instr	Operands	Effect
0	MOVE	src, dst	dst := src
1	ADD	src, dst	dst := dst + src
2	ADDC	src, dst	dst := dst + src + C
3	SUB	src, dst	dst := dst - src
4	SUBC	src, dst	dst := dst - src - C
5	SHL	src, dst	dst << src, fill with X, shift to C
6	SHR	src, dst	dst >> src, fill with C, shift to X
7	SWAP	src, dst	dst := ((src << 8) & 0xFF00) | ((src >> 8) & 0xFF)
8	NOT	src, dst	dst := !src
9	AND	src, dst	dst := dst & src
A	OR	src, dst	dst := dst | src
B	XOR	src, dst	dst := dst ^ src
C	CMP	src, dst	compare src with dst
D			reserved
E	HALT		Halt the processor
D	ABRA	dest, [!]cond	Absolute branch
D	ASUB	dest, [!]cond	Absolut subroutine call
D	RBRA	dest, [!]cond	Relative branch
D	RSUB	dest, [!]cond	Relative subroutine call

Jumps and Branches

4 bit	4 bit	2 bit	2 bit	1 bit	3 bit
opcode	src rxx	src mode	mode	negate condition	select condition

Addressing Modes

Mode bits	Notation	Description
00	Rxx	Use Rxx as operand
01	@Rxx	Use the memory cell addressed by the contents of Rxx as operand
10	@Rxx++	Use the memory cell addressed by the contents of Rxx as operand and then increment Rxx
11	@--Rxx	Decrement Rxx and then use the memory cell addressed by Rxx as operand

Shortcuts

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

Shortcut	Implementation
RET	MOVE @R13++, R15
INCRB	ADD 0x0100, R14
DECRB	SUB 0x0100, R14
NOP	ABRA R15, 1
SYSCALL(x, y)	ASUB x, y

sysdef.asm also defines three shortcuts SP, SR, and PC for R13, R14, and R15.

Input/Output

I/O devices are memory mapped, their respective control and data registers occupy the topmost 1 kW memory page.

Label	Address	Description
IO$BASE	0xFF00	Start of I/O area
VGA$STATE	0xFF00	VGA status register
VGA$CR_X	0xFF01	Cursor X-position
VGA$CR_Y	0xFF02	Cursor y-position
VGA$CHAR	0xFF03	Character code
VGA$OFFS_DISPLAY	0xFF04	Display RAM offset
VGA$OFFS_RW	0xFF05	R/W RAM offset
IO$TIL_DISPLAY	0xFF10	TIL-display
IO$TIL_MASK	0xFF11	Mask register
IO$SWITCH_REG	0xFF12	Switch register
IO$KBD_STATE	0xFF13	USB-keyboard state
IO$KBD_DATA	0xFF14	USB-keyboard data
IO$CYC_LO	0xFF17	Cycle counter low
IO$CYC_MID	0xFF18	Cycle counter middle
IO$CYC_HI	0xFF19	Cycle counter high
IO$CYC_STATE	0xFF1A	Cycle counter status
IO$UART_SRA	0xFF21	UART status register
IO$UART_RHRA	0xFF22	UART receive register
IO$UART_THRA	0xFF23	UART receive register

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.
2...0	Display color (RGB).

VGA$CR_X Set this register to the X coordinate for the next character to be displayed.

VGA$CR_Y Y coordinate for the next character to be displayed.