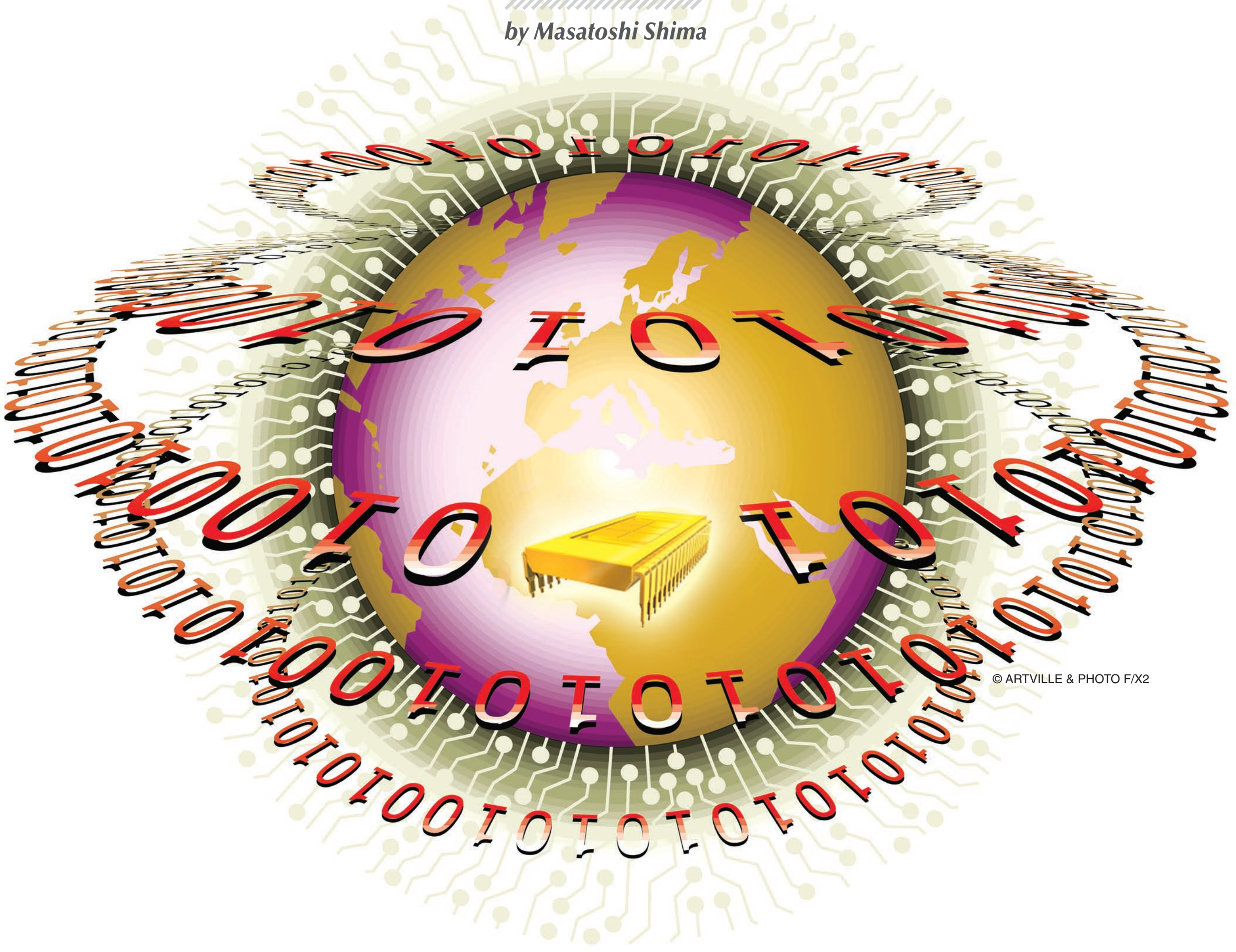
by Masatoshi Shima



The 4004 CPU of My Youth

Developing the world's first microprocessor.

This article is a recollection of the development of the world's first microprocessor, the 4004, as seen from Busicom Corp., the Japanese desktop calculator manufacturer where I was working from the late 1960s to the early 1970s. In 1969, Busicom Corp. launched a project to develop LSI chips for a ROM-based, macroinstruction-programmable decimal computer system. At that time, Busicom was a successful Japanese manufacturer of electronic calculators with a reputation for innovation. Through the LSI project, Busicom and Intel Corporation succeeded in March 1971 in developing the world's first 4-b microprocessor, the 4004, a product that was conceptually the exclusive property of Busicom.

I worked at Busicom from the late 1960s to the early 1970s to develop the 4004. In this article, I recall my role throughout the development process, including:

- the development of a printing desktop calculator using ROM-based programmed-logic that led to the birth of the 4004

Digital Object Identifier 10.1109/MSSC.2008.930946

We designed the logic at the transistor level instead of the gate level so that it could be used for both the circuit and the layout designs.

- clarifying problems with the binary processor proposed by Intel
- optimizing the 4004 system configuration and its instruction set
- designing the logic of the 4004 CPU
- developing the 4004-based printing desktop calculator.

The first transistor desktop calculator had been developed in Japan a few years earlier, in 1964. It was quickly put into commercial production, leading to a rapid growth in the calculator market and making Japan a major manufacturer of calculators. At that time, desktop calculators still used wired logic. For calculator technology to proceed, however, a new approach to logic design was needed—one that would allow the design, modification, and addition of new system functions in a short period of time.

Moving to ROM-Based Programmed-Logic

In 1968 at Busicom, I developed a printing desktop calculator with a ROM-based, macroinstruction programmable decimal computer architecture. Like a computer, the calculator consisted of:

- input-output (I/O) devices comprising a keyboard, a display, and a printer
- a processor for data processing
- ROM for program memory
- data memory.

Figure 1 shows a block diagram of the CPU in this calculator. The CPU was multichip; its data path consisted of a 4-b serial ALU operating on decimal and binary data, an accumulator, a keyboard input register, a multiplication and division register; control registers (comprising counter, decimal point register, and flag status register), a timing module; and data memory.

The control unit consisted of an 8-b program counter with an address incrementer, a ROM for storing the program, an 8-b instruction register, an instruction decoder, and an instruction execution control module. The instruction set included 23 instructions for operations such as data transfer, data exchange, load immediate, clear, addition and subtraction of decimal data, shift, addition and subtraction of binary data, increment and decrement, flag control, unconditional jump, conditional jump, and print. The program size was less than 256 B. The I/O device control was implemented in wired logic.

LSI Enters the Scene

In 1969, Busicom started development of LSI chips for a decimal computer system. First of all, the number

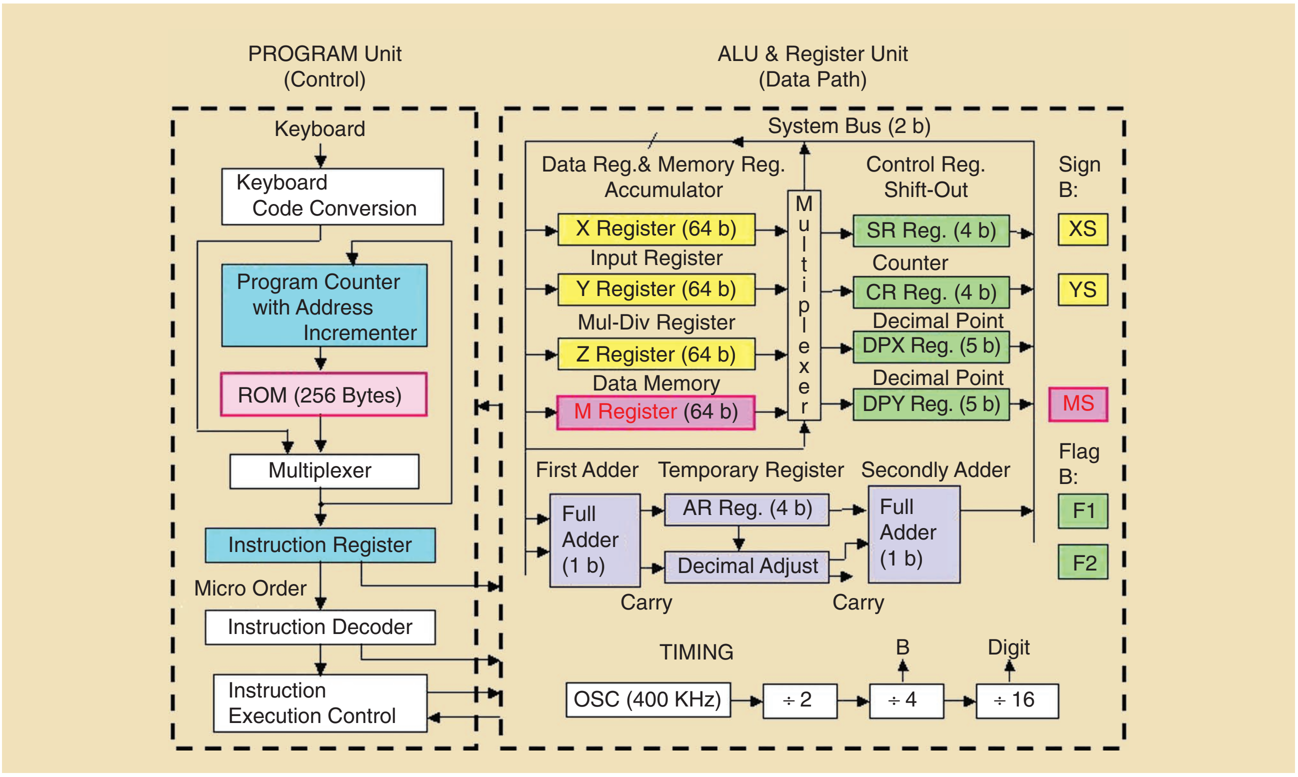
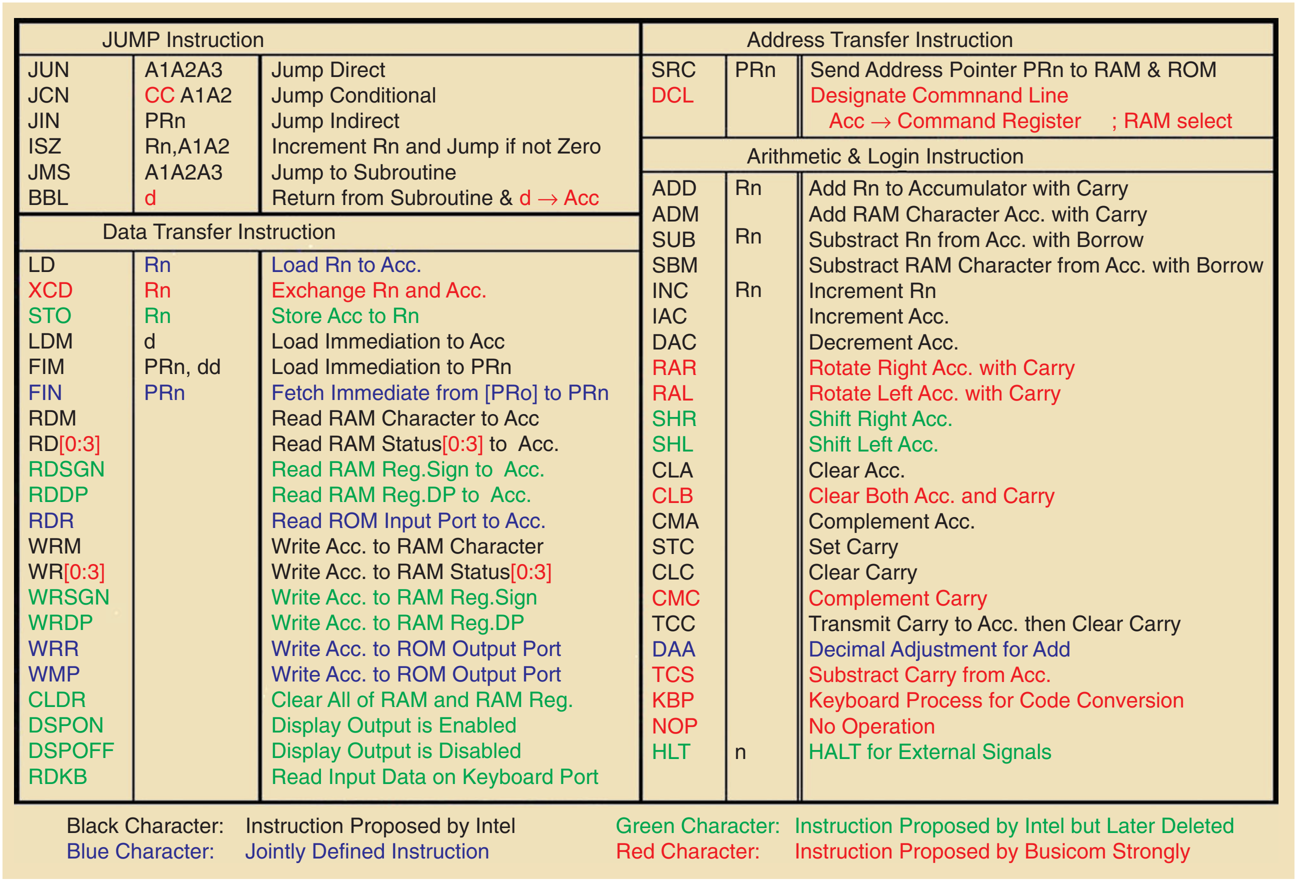


FIGURE 1: Block diagram of the CPU of a ROM-based, macroinstruction programmable desktop calculator.



JUMP Instruction		
JUN	A1A2A3	Jump Direct
JCN	CC A1A2	Jump Conditional
JIN	PRn	Jump Indirect
ISZ	Rn,A1A2	Increment Rn and Jump if not Zero
JMS	A1A2A3	Jump to Subroutine
BBL	d	Return from Subroutine & d → Acc
Data Transfer Instruction		
LD	Rn	Load Rn to Acc.
XCD	Rn	Exchange Rn and Acc.
STO	Rn	Store Acc to Rn
LDM	d	Load Immediation to Acc
FIM	PRn, dd	Load Immediation to PRn
FIN	PRn	Fetch Immediate from [PRo] to PRn
RDM		Read RAM Character to Acc
RD[0:3]		Read RAM Status[0:3] to Acc.
RDSGN		Read RAM Reg.Sign to Acc.
RDDP		Read RAM Reg.DP to Acc.
RDR		Read ROM Input Port to Acc.
WRM		Write Acc. to RAM Character
WR[0:3]		Write Acc. to RAM Status[0:3]
WRSGN		Write Acc. to RAM Reg.Sign
WRDP		Write Acc. to RAM Reg.DP
WRR		Write Acc. to ROM Output Port
WMP		Write Acc. to ROM Output Port
CLDR		Clear All of RAM and RAM Reg.
DSPON		Display Output is Enabled
DSPOFF		Display Output is Disabled
RDKB		Read Input Data on Keyboard Port

Address Transfer Instruction		
SRC	PRn	Send Address Pointer PRn to RAM & ROM
DCL		Designate Commnand Line Acc → Command Register ; RAM select
Arithmetic & Login Instruction		
ADD	Rn	Add Rn to Accumulator with Carry
ADM		Add RAM Character Acc. with Carry
SUB	Rn	Substract Rn from Acc. with Borrow
SBM		Substract RAM Character from Acc. with Borrow
INC	Rn	Increment Rn
IAC		Increment Acc.
DAC		Decrement Acc.
RAR		Rotate Right Acc. with Carry
RAL		Rotate Left Acc. with Carry
SHR		Shift Right Acc.
SHL		Shift Left Acc.
CLA		Clear Acc.
CLB		Clear Both Acc. and Carry
CMA		Complement Acc.
STC		Set Carry
CLC		Clear Carry
CMC		Complement Carry
TCC		Transmit Carry to Acc. then Clear Carry
DAA		Decimal Adjustment for Add
TCS		Substract Carry from Acc.
KBP		Keyboard Process for Code Conversion
NOP		No Operation
HLT	n	HALT for External Signals

Black Character:	Instruction Proposed by Intel	Green Character:	Instruction Proposed by Intel but Later Deleted
Blue Character:	Jointly Defined Instruction	Red Character:	Instruction Proposed by Busicom Strongly

FIGURE 2: History of the 4004 instruction set.

of control registers was increased to eight; these registers were collectively called the "index register." The instruction set was improved, and subroutine jump and processor stop instructions were added.

Busicom planned to develop seven different types of LSI chips, one each for the program control, the arithmetic unit, the ROM for program, the shift register for data, the timing circuit, the printer control, and the output buffer. The printing calculator was to be constructed with nine LSI chips. The chips were to be used in a variety of applications—business calculators, scientific calculators, billing machines, teller machines, and cash registers.

Busicom selected Intel as its development partner for this particular project because Intel had developed a high-performance and high-density silicon-gate MOS process. For confidentiality reasons, Busicom did not disclose to Intel its plans to use the chips in applications other than calculators.

With two other project members, Hiroyuki Masuda and Shogo Takayama, I visited Intel in California in June 1969 to work together on the development of the LSI. At that time, Intel was a small semiconductor company specializing in memory chips. Without logic designers, they did not have a clear picture of the logic used in our calculators and responded negatively to our proposal for developing a family of various LSI chips based on combinational logic and sequential logic. Fortunately for us, however, Marcian E. (Ted) Hoff, who was assigned to work for this project, showed interest in the ROM-based, macroinstruction programmable decimal computer system, the instruction set, and the calculator program that Busicom proposed for the project.

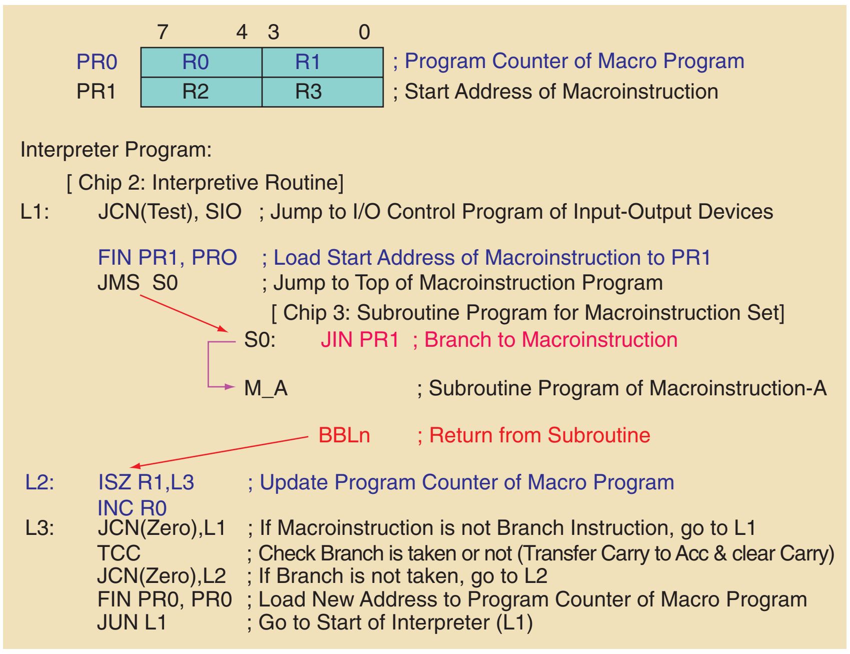
Intel's Proposal

It was in late August 1969, when the project was at a standstill, that

DECODING THE ACRONYMS

Many of the acronyms in this article are so familiar they hardly need definitions. But, for the record, here is a complete list of terms with their meanings.

LSI	large-scale integration
ROM	read-only memory
CPU	central processing unit
I/O	input-output
ALU	arithmetic logic unit
MOS	metal-oxide semiconductor
DAA	decimal adjust accumulator instruction
BCD	binary-coded decimal
DRAM	dynamic random-access memory
PLA	programmable logic array
RAM	random-access memory
SR	shift register
RISC	reduced instruction set computer



```
              7       4  3       0
PR0      |   R0   |   R1   |  ; Program Counter of Macro Program
PR1      |   R2   |   R3   |  ; Start Address of Macroinstruction

Interpreter Program:
     [ Chip 2: Interpretive Routine]
L1:     JCN(Test), SIO   ; Jump to I/O Control Program of Input-Output Devices

        FIN PR1, PRO     ; Load Start Address of Macroinstruction to PR1
        JMS  S0          ; Jump to Top of Macroinstruction Program
                           [ Chip 3: Subroutine Program for Macroinstruction Set]
                    S0:      JIN PR1   ; Branch to Macroinstruction

                    M_A                 ; Subroutine Program of Macroinstruction-A

                             BBLn       ; Return from Subroutine

L2:     ISZ R1,L3        ; Update Program Counter of Macro Program
        INC R0
L3:     JCN(Zero),L1     ; If Macroinstruction is not Branch Instruction, go to L1
        TCC              ; Check Branch is taken or not (Transfer Carry to Acc & clear Carry)
        JCN(Zero),L2     ; If Branch is not taken, go to L2
        FIN PR0, PR0     ; Load New Address to Program Counter of Macro Program
        JUN L1           ; Go to Start of Interpreter (L1)
```

FIGURE 3: The interpreter program.

Hoff proposed to us his basic idea of a binary processor. He showed us a block diagram consisting of

- a 4-b parallel ALU including a 4-b accumulator; 16 sets of 4-b index registers (which could also work in pairs as eight 8-b registers)
- a 4-level, 12-b address stack holding the program counter and three return addresses registers, allowing for three levels of nested subroutines
- keyboard input pins.

There were, however, some major problems with the proposed binary processor:

- It focused only on processor function.
- The arithmetic instructions operated only on binary data.
- It lacked several instructions necessary for interpreting application programs that use a macroinstruction set.
- It lacked instructions and features required to control I/O devices in real time.
- The proposed instruction set was too primitive.

In responding to Intel's proposal, I emphasized the need to meet the demands of the application in the system architecture and instruction set.

With this emphasis in mind, as well as our goal of using only LSI chips to construct a system, Busicom and Intel agreed to develop a program ROM and a data RAM in addition to the processor. Then, we jointly defined a 4-b time-multiplexed bidirectional system bus for connecting the processor directly with both ROM and RAM. We also agreed to add a decimal adjust addition (DAA) instruction to be used after the binary addition of binary-coded decimal (BCD) data in multidigit decimal numbers. With DAA, the time required for one-digit of addition was reduced to 107 microseconds with a 750-kHz clock.

I conducted both a static evaluation and a dynamic evaluation of the proposed instruction set by utilizing a macroinstruction set and an application program such as would be used in the printing calculator. I also asked Busicom to measure the contact time of the keyboard's key. On the basis of these evaluations and measurement, I concluded that it would be critical to the success of the project to add a collection of new useful instructions, redefine some instructions, and delete unnecessary instructions; with these changes it would be possible to reduce the ROM size, increase performance, implement an interpreter, and achieve real-time control of I/O devices. Figure 2 shows the sources of the instructions in the final 4004 instruction set.

Developing the Instruction Set

The instruction development proceeded in the following sequence. For real-time control of I/O devices, I added the status of an external input pin (Test) as one of conditions of the conditional jump instruction (JCN). Next I added a 10-b static shift register LSI (SR) to serve not only for keyboard scanning but also as an output buffer for a printer.

Further, I added the keyboard process (KBP) instruction for converting a 4-b row data of the keyboard into 0, 1, 2, 3, or 4, or, if two or more keys were pressed simultaneously, into 15.

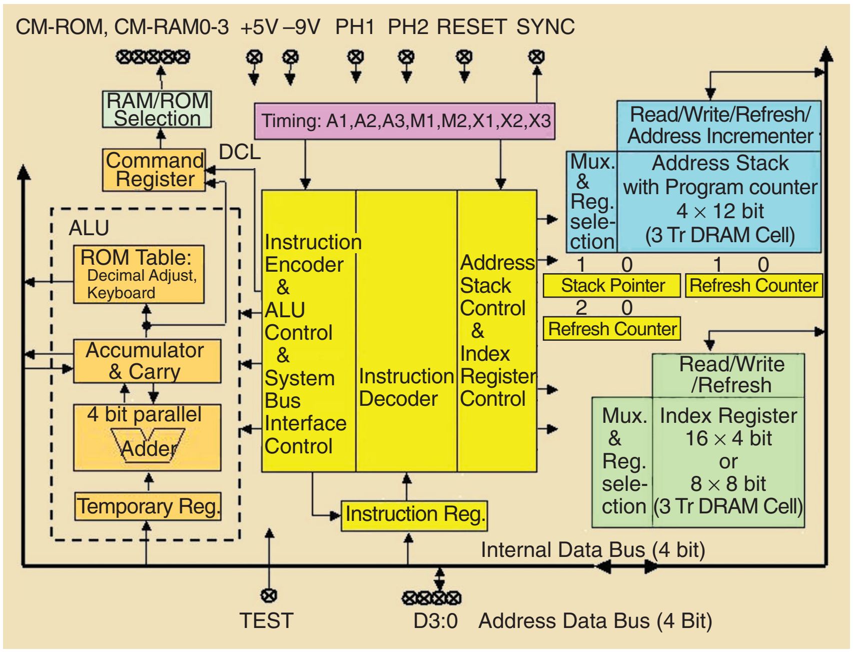
So that we could implement an interpreter program, we added a fetch indirect instruction (FIN) to load the 8-b data from ROM addressed by the pair register PR0 into the pair register PRn. Next I redefined the BBL instruction (branch back and load data to the accumulator) so that it loads a constant return value *n* into the accumulator on execution.

Figure 3 shows the interpreter program. It uses PR0 as the program counter of the macro program while the PR1 register holds the start address in ROM for the macroinstruction to be executed. The interpreter first executes JCN, which checks the Test input to determine if there is a request from an external device. If there is, it jumps to the I/O control program; if not, the interpreter begins normal program execution.

First, the interpreter executes FIN to fetch the start address for the current macroinstruction from the ROM addressed by PR0 and stores this start address into PR1. Next, using the jump to subroutine instruction (JMS), it transfers execution to the top of the macroinstruction program; then it jumps to the current macroinstruction's program by using the register-indirect jump instruction (JIN) with PR1.

In the last step of execution of the macroinstruction program, a nonzero return value is stored into the accumulator with BBL if the macroinstruction is a branch instruction, and the carry is set if the branch condition is not met.

On returning from the macroinstruction program, the interpreter updates PR0, processes the branch macroinstruction if any, and then returns to its start. The program size of interpretive routine in the interpreter was 16 B, the longest macroinstruction took 2 ms to execute, and real-time control of I/O devices was made possible by polling the synchronization signal of the printer via the Test input pin at 2-ms intervals in the interpreter. This became one of the keys to the success of the 4004 system.



CM-ROM, CM-RAM0-3 +5V –9V PH1 PH2 RESET SYNC
RAM/ROM Selection
Timing: A1,A2,A3,M1,M2,X1,X2,X3
Read/Write/Refresh/ Address Incrementer
Command Register
DCL
Mux. & Reg. sele-ction
Address Stack with Program counter 4 × 12 bit (3 Tr DRAM Cell)
ALU
ROM Table: Decimal Adjust, Keyboard
Instruction Encoder & ALU Control & System Bus Interface Control
Address Stack Control & Index Register Control
1 0
1 0
Stack Pointer
Refresh Counter
2 0
Refresh Counter
Accumulator & Carry
Instruction Decoder
Read/Write /Refresh
4 bit parallel Adder
Mux. & Reg. sele-ction
Index Register 16 × 4 bit or 8 × 8 bit (3 Tr DRAM Cell)
Temporary Reg.
Instruction Reg.
Internal Data Bus (4 bit)
TEST
D3:0 Address Data Bus (4 Bit)

FIGURE 4: Block diagram of the 4004 CPU.

Optimizing Instructions

The original proposal from Intel included the store instruction (STO) to transfer data from the accumulator to the index register, but it did not include the load instruction (LD) to transfer the data in the opposite direction. In optimizing the instruction set, the STO instruction was replaced with LD, and the exchange instruction (XCH), which swaps the contents of accumulator and index register, was added. The XCH instruction does not destroy the contents of either register. The addition of XCH and eight more RAM status registers per chip made up for the lack in number of registers in the index register.

The proposed shift instruction (SH) was replaced with the more useful rotate instruction (RA). To reduce program size further, the following instructions were also added: the clear-both instruction (CLB) to clear both the accumulator and the carry; the complement-carry instruction (CMC); the transfer-carry-subtract instruction (TCS) to be used for decimal subtraction; and the no-operation instruction (NOP) to be used for the software timer and the debugging. Finally, the designate command line instruction (DCL) and the chip select output pins were added for selecting RAM without an external decoder circuit. At the same time, keyboard input pins and the halt instruction (HLT) were deleted.

It took me three months to optimize the system configuration and the instruction set. However, with these major changes to the instruction set, I reduced program size by around 30%.

A Brief Return to Japan

I returned temporarily to Japan in December 1969 to finalize the details of the printing calculator's program and confirm the instruction set. Busicom and Intel formally signed a contract for the development of the LSI chips on February 6, 1970, with the development fee set at US$60,000. In mid-March of that year, Busicom sent Intel the formal functional specification and the instruction set, attaching detailed diagrams to avoid misunderstanding.

I visited Intel again to verify the logic in April 1970, only to find that the project had made little progress since I had returned to Japan—all Intel had done in the intervening months was to hire one development engineer and two layout designers for the project. There was no logic design engineer. Consequently, I joined the design group and took charge of the logic design of the 4004 CPU, logic simulation, layout checking, and test program generation. Busicom in Tokyo took charge of building a CPU emulator for logic verification.

I designed the logic at the transistor level instead of the gate level so that it could be used for both the circuit and the layout designs. First, I clearly defined the interface signals between the functional modules and then made a detailed functional block diagram of the 4004 CPU before proceeding with the detailed logic design. Figure 4 shows the block diagram of the 4004 CPU. This was an important step; back in the 1970s, the success of a microprocessor development depended largely on the quality of its detailed block diagram.

Reducing the Transistor Count

Because we chose a three-transistor DRAM cell for the address stack and the index register, each cell contained one read bit line and one write bit line. First of all, an address incrementer was built in the refresh circuit of the address stack and was executable in one state by using two interleaved clocks. Next, in order to calculate early the total number of transistors needed, the logic design started from those modules that used relatively large number of transistors. Those modules were the index register (402 transistors), the address stack with an address incrementer (374), the instruction register (72), the timing circuit (99), the command control (100), and the system bus interface control (92), which brought the total number of transistors used to 1139. This meant that it would be impossible to design the 4004 CPU within the 2,000-transistor limit that was first estimated by Intel in the proposal.

The logic design was now reaching its completion. In order to simplify the logic—and thereby reduce the transistor count—we decided that any machine cycle would be composed of eight states (A1, A2, A3, M1, M2, X1, X2, and X3). A combinational logic circuit was used for instruction execution control, and constructed with signals of the instruction decoder (48 PLA terms), the instruction encoder (6 terms), state timing, and machine cycle timing. A major challenge at this point—and critical for logic simplification—was optimizing by balancing the selection of the instruction decoders and encoders with the number of combinational logic circuits.

Next, taking the overall layout into consideration, I carefully designed the ALU module (329 transistors), including the accumulator and decimal adjust circuit, the instruction decoder and encoder (269), and the instruction execution control module (500). Then, to reduce testing time, I decided to send the accumulator and the carry flag out to the system bus when it was idle. In the end, the number of transistors used for the 4004 totaled 2,237.

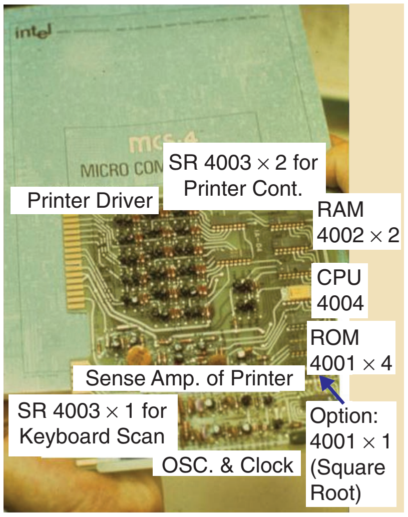

Printer Driver
SR 4003 × 2 for Printer Cont.
RAM 4002 × 2
CPU 4004
ROM 4001 × 4
Sense Amp. of Printer
SR 4003 × 1 for Keyboard Scan
Option: 4001 × 1 (Square Root)
OSC. & Clock

FIGURE 5: Motherboard of the 4004 microprocessor-based printing desktop calculator.

Busicom selected Intel as its development partner for this particular project because Intel had developed a high-performance and high-density silicon-gate MOS process.

Only One Logic Error

In the logic simulation carried out in August 1970, only one logic error was found. Consequently, the layout of the 4004 was almost the same as the transistor-based logic circuit schematics that I drew in the last phase of the logic design. I have been feeling that after Busicom transferred the rights to the 4004 to Intel in 1974, Busicom's contribution to the 4004 development faded out.

Back to the Desktop Calculator

After completing the test program generation and the layout checking, I flew back to Japan in November 1970 and returned to my original work of developing the printing desktop calculator. At Busicom, I built an engineering prototype with the 4004 CPU, 4001 ROM emulator, 4002 RAM, 4003 SR, a card reader, and a control panel with display. The program for the calculator was completed in March 1971. The total program size was kept to within 1,000 B, with approximately 250 B allocated for the application program, 400 B for the macroinstruction program, and 350 B for the interpretive routine, keyboard control, and printer control.

Figure 5 shows the motherboard of the 4004-based printing desktop calculator. Figure 6 shows the world's first microprocessor-based calculator.

The Moment of Truth

The long-awaited moment finally came in April 1971 when the 4004 CPU arrived at Haneda Airport. After some simple testing of the 4004 CPU, the 1-Kbyte program was loaded into the ROM emulator via a card reader. Now all I had to do was push and release the reset button of the engineering prototype. While not being afraid of failure, I was also fully aware that the outcome of the two-year project would be determined in that one moment. I pushed the reset button, but hesitated for a while before releasing it; once released, the final result would be inescapable.

After a deep breath, I took the plunge and released the button. The calculator activated, and the program started to run. The address display of the program counter indicated that the keyboard scan program was executing. Unable to wait any longer, I pressed the number

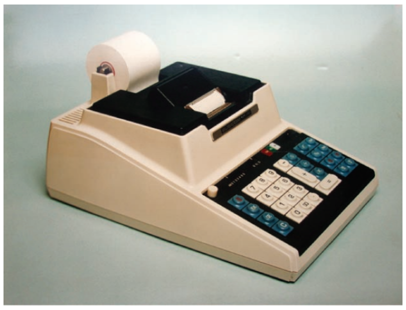


FIGURE 6: The world's first microprocessor-based calculator.

keys and then the addition key. It felt like I waited forever for the printer to start. Finally, the printer roared into life, and the numbers and addition symbol I had input were printed out. I pressed some more number keys, the addition key, and the equal key at the end. "It's working!" I felt my heart pounding and my entire body flash hot with excitement, while my head alone remained sober.

This was the very moment of the birth of the world's first microprocessor, the 4004, and the microprocessor-based system. I still remember vividly the feel of the reset button on the prototype model, the hopes and fears I felt, and my racing heartbeat when I released that button.

More Innovative Microprocessors

After the 4004-based calculator had been successfully developed, I moved to Ricoh Corp. There, using NEC's NEAC-M4 8-b minicomputer, I designed a controller for a high-speed printer and a production tester for a drum memory. Then I joined Intel in 1972 and developed the 8-b 8080 microprocessor, as well as the 8080 peripheral LSI chips as supervising engineer (Figure 7). I moved on to Zilog and developed the Z80, which later came to be called the ultimate 8-b microprocessor. I returned to Japan in 1980 after completing the development of the 16-b Z8000 microprocessor.

My life as a development engineer was enriched by my decade-long active involvement in developing innovative microprocessors, and I look back with nostalgia on those exciting pioneering days.

The Role of Applications

During that decade, I became convinced that many innovative products were developed where the new generation's architecture was required in the application, and at the same time it has been quite

FIGURE 7: Masatoshi Shima in 1973 at Intel during development of the 8080 microprocessor.

important to optimize between the product's specification and its implementation. After establishing the Intel Japan Design Center in 1980, I developed application-specific microprocessors such as:

- an Intel 8051–based multi-microprocessor, composed of one main processor and ten processor elements, for a copier application in 1985 at Intel Japan
- an x86-compatible low-power microprocessor series for a Japanese word processor application in 1988 at VM Technology Corp.
- a 32-b RISC processor–based multi-microprocessor for a document processing system, such as a digital copier in 1999 at TOPS Corp.

My life as a development engineer was enriched by my decade-long active involvement in developing innovative microprocessors.

By the time I had acquired a doctor of engineering at Tsukuba University in 1992, I clearly understood the things that were important for me in microprocessor development: the instruction set, the instruction decoder, and the hardware architecture that inlcudes internal bus architecture. In 2000, I moved to Aizu University for teaching and researching computer architecture, designing pipeline microprocessors and a Java just-in-time-compiler, and teaching the history of the microprocessor from the viewpoint of a microprocessor development engineer.

About the Author

Masatoshi Shima (Shima80z@green.ocn.ne.jp) is a former professor at Aizu University in Japan. Previously, he was manager of the Intel Japan Design Center. While an employee at Busicom Corp. in Japan, he worked as a programmer of the Mitsubishi-Melcom-3100 computer and as a logic designer for desktop calculators. In 1968, he developed the printing desktop calculator, in which he introduced ROM-based stored programming technology with decimal-plus-binary computer architecture.

After working on the development of the 4000 series of microprocessors, he joined Intel, where he developed the 8080 and several peripheral chips as supervising manager. As manager of high-end microprocessors at Zilog, he developed the Z80 and Z8000. He received a B.S. in chemistry from Tohoku University in 1967 and a doctor of engineering degree from Tsukuba University in 1992, Japan. He published *The Birth of the Microprocessor: My Recollection*, (Tokyo: Iwanami Shoten Publishing) in 1987. He received the Kyoto Prize in 1997, the "Inventor of the Microprocessor Unit" Award in 1998 at the 50th Anniversary of the Semiconductor Industry, and the Funai Achievement Award at the Forum on Information Technology in 2006. SSC