IETE Journal of Education

ISSN: (Print) 0974-7338 (Online) Journal homepage: http://www.tandfonline.com/loi/tije20

32-Bit Microprocessors

R. Govinda Rajulu

To cite this article: R. Govinda Rajulu (1986) 32-Bit Microprocessors, IETE Journal of Education, 27:1, 18-34, DOI: 10.1080/09747338.1986.11436094

To link to this article: http://dx.doi.org/10.1080/09747338.1986.11436094

Published online: 02 Jun 2015.

32-Bit Microprocessors

(Review Paper)

R. GOVINDA RAJULU

Based on the literature available, the architectural features of 32-bit Microprocessors are described. The processors considered are NS 16032, Bellmac-32, HP 32-bit CPU and iAPX 432.

THE microprocessor revolution is proceeding at a pace perhaps unparalleled in scientific history. Since the introduction of the first computer on a chip in 1971, we have seen four generations of microprocessors, and the number of devices per chip has increased by a factor of 200, the clock frequency by a factor of 50, and the overall throughput of the microprocessor has increased by two or three orders of magnitude. The pace of this revolution even outstrips the pace of advancement of computer. In just a little over a decade, microprocessors have evolved from the Intel 4004, a four-bit machine, to 32-bit chips. The 4-bit 4004 was introduced in 1971. Successive generations of microprocessors using 8-bit, 16-bit, and 32-bit chips commenced in 1972, 1974 and 1981, respectively. The characteristics and performance of single-chip 16-bit microprocessors have been summarized earlier [6, 7, 8].

Microprocessors with 32-bit internal paths and 16-bit external paths have been in existence since 1980. We discuss general trends by considering the following 32 bit microprocessors chosen on the basis of their architectural innovativeness and the availability of good documentation. These are the NS 16032 chip designed by National, the Bellmac-32A from Bell Labs, the no-name 32-bit CPU chip from Hewlett-Packard, and the Intel iAPX 432. The general characteristics of the four microprocessors are summarized in Table 1.

The NS 16032

This microprocessor represents the high end of the NS16000 family. Internally it uses 32-bit paths and 32-bit address arithmetic to access memory without the overhead of segmentation registers. Externally, it uses a multiplexed address and data bus with 24 bits of external address and 16-bits of external data. This multiplexing has enabled the device to be packaged in a 48-pin package, at the cost of reduced external data rates. The NS 16032 chip implements 82 basic instructions and has an address range of 16 M-bytes with 25-bit address pointers. This address range can be enhanced to 32 M-bytes with a memory management unit. The chip has a three stage pipe-line. Implementation of an eight-byte instruction fetch-ahead queue provides an overlapped instruction fetch and execute facility. Other highlights include modular software capabilities, a fully

Manuscript received 1984 Novembr 11, revised 1985 April 4.

*Department of Computer Science, Regional Engineering College, Warangal 506 004.

Table 1 [7] *General characteristics of the four 32-bit microprocessors*

	NS 16032[15]	*BELLMAC-32A*[16,24]	*HP 32-BIT CPU*[17]	*INTEL iAPX 432*[18]
Year of Commercial Introduction	1982	1982	1982	1981
Technology	3.5μm NMOS	2.5μm domino CMOS	1.5/1.0μm NMOS	HMOS
No. of transistors	60,000	146,000	450,000	219,000 on 3 chips
Size of chip	84,000 mil^2	160,000 mil^2	48,400 mil^2	100,000 mil^2 each
Power dissipation	1.25 W	0.7 W at 8 MHz	4 W	2.5 W/chip
Pin count	48	63 active 84 total	83	64 per chip
Basic clock frequency	10 MHz	10 MHz	18 MHz	8 MHz
Direct address range (Bytes)	2^{24}, 2^{25} with MMU	2^{23}	2^{29} real; 2^{41} virtual	2^{24} real; 2^{40} virtual
No. of general-purpose registers	8	16 user-visible	28 (not all general-purpose)	No registers visible to user
No. of basic instructions	82	169	230	221
No. of addressing modes	9	18	10	5

orthogonal instruction set, a virtual memory facility, and string processing capabilities. These features can be expanded through the use of auxiliary chips. The NS 16081, for example, provides 32-bit and 64-bit floating-point capabilities, while the NS 16082 memory management unit supports demand paged virtual memory. Communication between the NS 16032 CPU and the NS 16202 interrupt control unit may involve transfer of information on the local bus as well as on the system bus, and incur the overhead and delays inherent in such transfers.

The basic structure of the NS 16032 is shown in Fig 1. The instructions are fetched via the 24-bit-wide multiplexed address/data bus. An instruction fetch-ahead queue provides overlap of instruction fetch and execution and aligns incoming instructions to 16-bit boundaries. The queue consists of double-ended eight-byte FIFO with 16-bit input and output buses. The loader extracts instruction from the queue and decodes it into a basic opcode, an operand length, address modes, and ALU control fields. The loader can decode two bytes every 100 nano seconds so that complex instructions can be

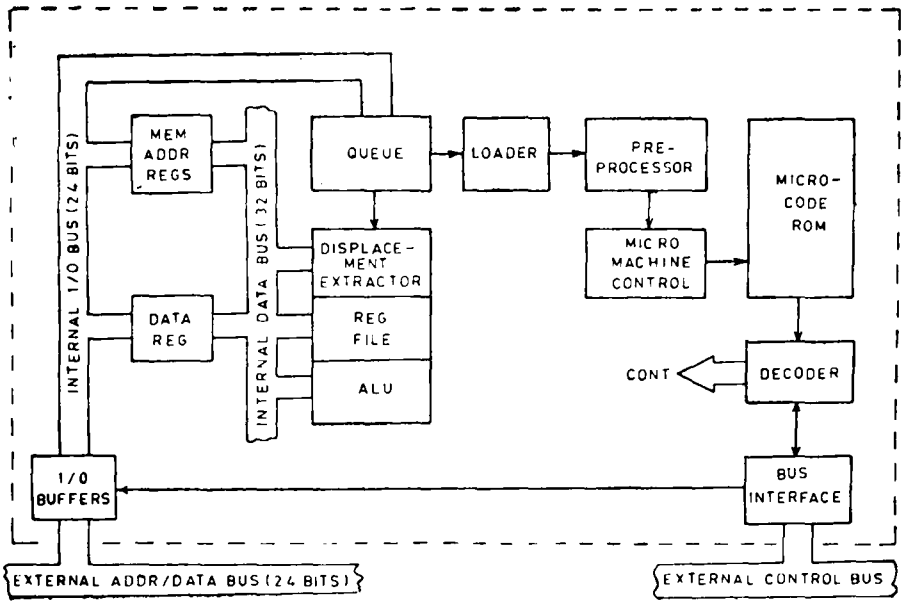

Fig 1 NS 16032 block diagram

efficiently executed without consuming excessive microcode ROM spaces. A two-level microcode enables sharing of common effective address-calculation routines by all instructions and avoids both time-consuming subroutine calls and space-consuming repeated microcode flows. A preprocessor controls the sequence of instruction execution, while the micromachine controls the individual execution steps. As microinstruction is executed, microinstruction $n+1$ is decoded, and microinstruction $n+2$ selected by the ROM address decoder. The procedure results in an effective 100-nanosecond microinstruction execution rate. A hardware backup mechanism saves contents of the processor status register, the stack pointer, and the program counter, and automatically restores them to their original values in the case of an instruction retry.

The NS 16032 contains sixteen registers-eight general-purpose ones (all 32-bit wide) and eight dedicated ones. The dedicated registers are as follows:

* The program counter register (24 bits) points to the current instruction being executed.
* The frame pointer register (24 bits) is used to access parameters and local variables on the stack.
* The static base register (24 bits) is used by software modules as a pointer to their respective global variables.
* The interrupt register (24 bits) points to the despatch table for interrupts and traps.
* The two stack pointer registers (2 pointers each, 24 bits wide) point to the top of the stack used for interrupt routines and to the top of the stack used for other programs, respectively.

The processor status register contains status code. The module register points to the module description of the currently executing software module. The module register and the processor status register together use 32-bits.

The original specifications of the NS 16032 were too ambitious to be accommodated on a single chip with current technology. Several functions were off-loaded from chip to auxiliary chips or slave processors. The total number of instructions was reduced from 100 to 82. Floating-point instructions are trapped by the main processor and executed by the NS 16081 floating-point unit. The NS 16032 processor accepts data in bits, bytes, half words and words. Its ability to operate on variable length character strings suits it to text processing applications. The NS 16032 offers instructions to operate on packed decimal quantities, on arrays and blocks. Instructions are of variable length. Common zero-operand instructions, such as the branch instruction, have a one-byte opcode. One-operand and two-operand instructions use a two-byte instruction. All instructions can use all applicable addressing modes. For code compactness, instructions are aligned on byte boundaries rather than on half-word or word boundaries. To reduce register allocation problems, the instruction set has been designed to be symmetric with respect to memory and general register references.

The NS 16032 provides a 16 M bytes uniform address space and following nine types of address modes:

* Register addressing
* Immediate addressing
* Absolute addressing
* Register-relative addressing
* Memory-space addressing. This allows relocatable reference to memory areas commonly used in high-level languages
* Top-of-stack addressing.
* The memory-relative addressing mode. This is useful for handling address pointers and manipulating fields in a record.
* The scaled index addressing mode. This is quite useful for indexing into an array while the basic addressing mode points to the head of the array.
* The external addressing mode. This mode is used to reference operands external to the current executing module and allows module to be relocated without linkage editing.

The lengths of address-mode offset constants are encoded in the upper two bits of the offset so that small offsets (-64 to 63) will require only one byte instruction stream and larger offsets will take two or four bytes. The integration of the memory management unit (Fig 2) provides operating system and virtual memory support and enables addressing upto 32 M bytes. Virtual-address-to-real-address translation is accomplished through two levels of page tables and offset specifications. Each page, 512 bytes in size, is assigned a protection code, and this provides an access control mechanism. The memory management unit has eight registers and provides a flow tracing facility for both sequential and non-sequential instructions.

The 32-bit version, the NS 32032, uses the same architecture (of NS 16032) but provides 32-bit external data paths like the Motorola MC68020.

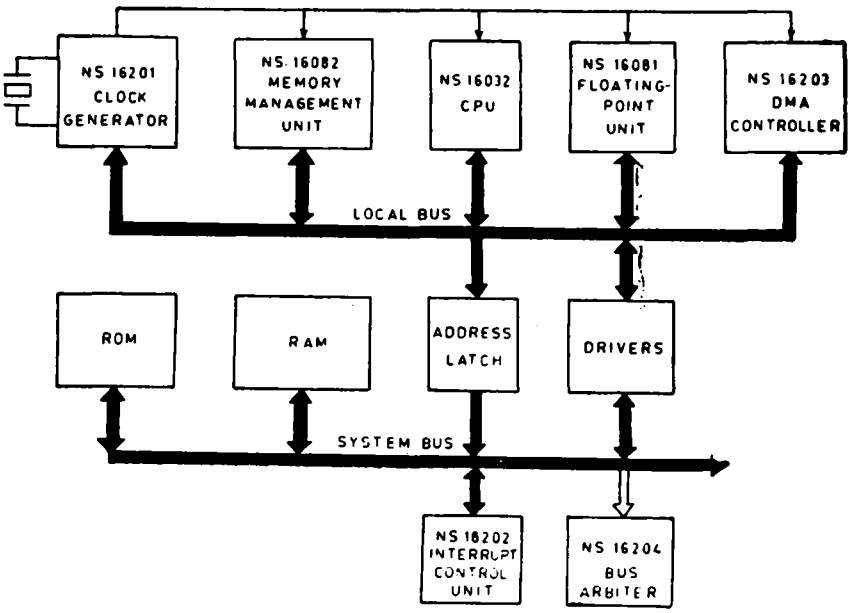

Fig 2 NS 16032 system

The Bellmac-32 A

Besides being a true 32-bit microprocessor, the Bellmac-32 is much more sophisticated technologically than the NS 16032. In general, CMOS circuits provide faster speed and lower power consumption than circuits implemented with traditional PMOS and NMOS technology. The Bellmac-32A single-chip CPU is fabricated in twin-tub CMOS and uses domino circuits that operate at twice the speed of previous CMOS circuits and enable a single clock pulse to activate many circuits simultaneously. The chip uses two non-overlapping clocks, each of 10 MHz and both generated from a 40 MHz external clock. The Bellmac-32A chip has been designed to provide support for the C programming language. Single instructions can move blocks of data from memory to memory, or push and pop a group of registers with respect to the stack. Sophisticated hardware facilities include a barrel shift circuit that shifts 0 to 31 bits in a single cycle. The operating system, which can be included in the address space of every process, includes a hardware interface to assist process-oriented operating system control software. It also includes a set of exception handling mechanisms. The exception structure provides four levels of execution privilege and intended for real-time control applications. Neither floating-point nor decimal arithmetic is supported although an auxiliary processor to perform such operation is being investigated. There is little compatibility with any existing microprocessor.

The Bellmac-32A chip was developed in a relatively short time through the extensive use of computer-aided design techniques. These techniques make the Bellmac-32A technology updatable. With these techniques, an existing desing can benefit from the advances in fabrication technology that permit thinner line

widths. Because of its CMOS implementation, the Bellmac-32A consumes the least power of the four microprocessors. The twin-tub CMOS process provides high switching speeds in both *n*-and *p*-channel devices because each tub is separately implanted for optimum doping. Classical CMOS logic designs have an equal number of *n*-and *p*-channel devices. Newer CMOS designs use 80 per cent *n*-channel devices and only 20 per cent *p*-channel devices. This ratio of *n*-channel to *p*-channel devices makes it possible to have high circuit densities while retaining the CMOS advantages of fast speed and low power consumption.

The structure of the Bellmac-32A is shown in Fig 3. Like the NS 16032, it consists of two distinct functional units: a fetch unit which controls interactions with the external memory and an execution unit which controls the manipulation and processing of data. Both units, as well as the bus, have full 32-bit capability. The instruction stream is byte-oriented. The first byte specifies the addressing mode and the register, and the subsequent bytes specify additional data. All bytes and half-word operands are sign-or zero-extended to 32-bits when they are fetched. Instructions fetched from memory are stored in instruction queue and translated into a series of microinstructions via PLA An arithmetic address unit performs all addr‹ss calculations. An ALU in the execute block performs the actual execution of microinstructions. The emphasis on support of pocess-oriented operating systems generates a need to store instructions, data and register values associated with a process whenever there is a switch from one process to another. The Bellmac-32A provides these storage functions in hardware.

The Bellmac-32A microprocessor contains a special program counter register and fifteen other registers, each 32 bits wide, that can be referenced in any addressing mode. Of these fifteen registers, three are used to support operating system functions (interrupt stack

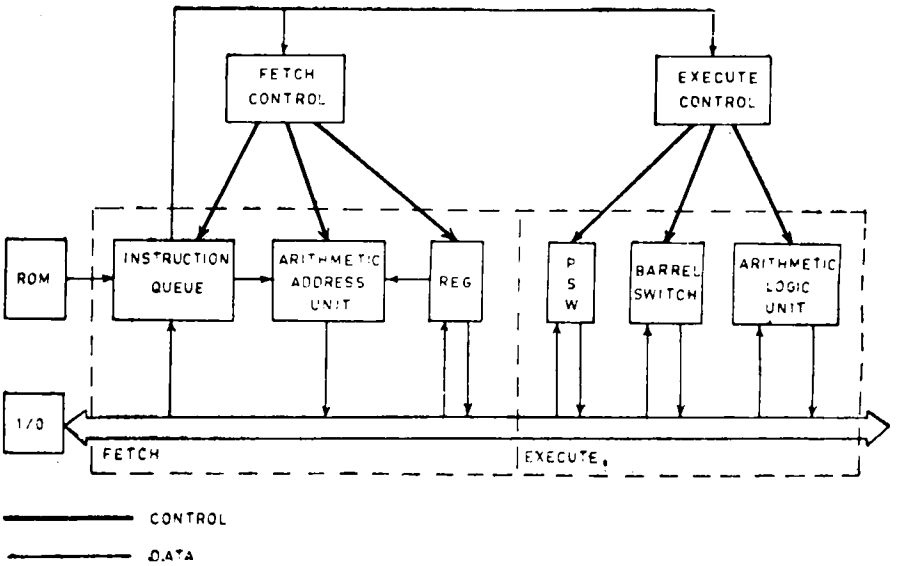

Fig 3 Bellmac-32 A CPU architecture

pointer, process control block pointer, processor status word) and can be written when the processor is in kernel execution level. Another three registers are used by certain instructions as a stack pointer, a frame pointer and an argument pointer.

The Bellmac-32A offers 169 instructions. It too supports bytes, half-words, words, and bit fields. Strings are supported by special block instructions, and the string format conforms to the C language. C compatibility is manifest in the implementation of the instruction repertoire. The result of the unary operations NEGATE and COMPLEMENT (implemented as move instructions) can either replace the existing datum or be placed in a new location. The dyadic form stores the result in the second operand and the triadic form places the result in the third operand, with the first two unaltered, these dyadic and triadic instructions are available for all operations. High level procedure linkage operations assist in manipulating the stack frame, saving registers, and transferring control between procedures. Also explicit instructions that permit the operating system to switch processes (CALL PROCESS and RETURN TO PROCESS) are provided. On the negative side Bellmac-32A does not support floating-point or decimal arithmetic.

The Bellmac-32A chip offers several addressing modes, literal, byte/half-word/word immediate, register, register deferred, short offset (for frame and argument pointers), absolute, absolute deferred, byte/half-word/word displacement deferred; and expanded operand. The Bellmac-32A chip includes a set of exception handling mechanisms and a hardware interface for a process-oriented operating system. This operating system can be included in the address space of each process, enabling each to execute independently.

The HP 32-Bit Chip

This no-name, device has the highest circuit density of any microprocessor-450,000 transistors on a single, 48,400 mil² chip. Implemented in double-layer-metal NMOS with a one-micrometre pitch, this microprocessor uses two non-overlapping clocks, each of 18 MHz frequency and both generated from an external 36 MHz clock.

It consists of seven major sections including:

(*i*) a microcode control store ROM which is organized as 9216 38-bit words,

(*ii*) a PLA for decoding microinstructions,

(*iii*) a sequencing machine for controlling the flow of instructions from the ROM to the PLA,

(*iv*) a test condition multiplexer for decoding various test and branch conditions,

(*v*) an extensive set of 32-bit registers for storage and manipulation of data,

(*vi*) a general purpose ALU, and

(*vii*) a memory-processor bus (MPB) interface for communication with the external data paths.

The chip is microcoded with 9 K (38 bit) words of ROM control store addressed via a set of 14-bit registers in the sequence stack. The microinstructions are decoded by a PLA. Most of the microinstructions execute in one clock cycle of 55 nanoseconds. Pipe-lining of memory operations permits initiation of a 32-bit memory read every two states (every 110 nanoseconds), eventhough the memory access time is longer. Like the other chips considered, the HP device has a three-level pipe line. This is implemented through a CIR (current instruction register) and NIR (next instruction register), and a PIR (prefetch instruction register). A self-test routine, executed by the CPU during powerup, automatically tests operations internal to the chip.

The HP chip offers several sophisticated hardware and software features. A hardware implemented n-bit barrel shifter can shift a 32-bit quantity right or left 0 to 31 placas in a single clock cycle. The load instruction includes automatic bounds checking and takes only 550 nanoseconds. The arithmetic and logical instructions can manipulate the 32-and 64-bit floating-point operands specified in the proposed IEEE floating-point standard. Text editing capabilities are inherent in the move and string instructions that manipulate byte arrays and string type data. All communications to and from the chip uses the memory-processor bus. This 32-bit-wide multiplexed address/data bus permits pipelined data transfers at 36 M bytes per second. Facilities to communicate with the memory and with peripheral devices are provided by a memory controller chip and an I/O processor chip, respectively.

The HP 32-bit chip is similar in operation to the previous two chips. In too uses a three stage overlapping sequence for instruction prefetch and execution and it too uses a microcoded structure. The microcode control-store ROM is organized as 9216 words, each of 38 bits. Microinstructions accessed from the ROM are decoded by a PLA. They drive control lines which determine the operations of the 32-bit register stack and the ALU. The flow of instructions to the PLA is controlled by a sequencing machine, which contains a microprogram counter, a set of incrementers, three registers for microcode subroutine return addresses, and a machine instruction opcode decoder. The opcode decoder generates the starting address in the control store for the microcode routine that implements each machine instruction. The test condition multiplexer facilitates conditional jumps and skips in the microcode. The ALU contains an n-bit shifter, a 32-bit logical selector, and a 32-bit full look-ahead adder which also performs integer multiplication and division via special hardware.

At the heart of the HP chip is a register stack, with 28 identical general-purpose registers, and an ALU, with four operand/result registers. Each of the general purpose registers is 32-bits wide; not all are accessable by software. The register stack uses two data buses and contains auxiliary logic such as top-of-stack and instruction registers. In any register, each of the 32 bit-cells can receive data from, or dump data to, either of the two data buses, as determined by the PLA outputs.

The HP 32-bit chip offers 230 instructions as well as 32-bit and 64-bit floating-point arithmetic. The load and store instructions can transfer double-words in addition to bits, bytes, half-words, and words. MOVE and STRING manipulate both unstructured byte arrays and structured string data. Hardware support includes four top-of stack registers to handle push and pop operations and to provide data valid indications. The large number of transistors enables hardware implementation of features that have traditionally been done by software. For example, run-time bounds checking of addresses, performed on all memory accesses, is supported in hardware.

The HP 32-bit processor views the memory space for each program as an active code segment (one of 4096 code segments), a stack segment, a global data segment, and a set of 4096 external data segments. Segment pointers, maintained in 32-bit on-chip registers, include base and limit registers for the code, stack, and global data segments, the current instruction address in the code segment, the address of the most recent stack marker in the stack segment, and the address of the top-of-stack in memory. External data segments are accessed via a set of memory resident tables. A memory controller chip can control upto 20 RAM chips,

each of 128 K bits, or upto eight ROM chips, each of 640 K bits, providing an effective memory space of upto 256 K bytes of RAM or 512 K bytes of ROM (Fig 4). The processor-memory bus has a transfer rate of 36 M bytes per second, and the overlapped access method permits high throughput. The memory controller maps logical to physical addresses in 16 K byte blocks, and permits byte, half-word, word and semaphore operations.

The iAPX 432

In the words of Myers iAPX 432 might be considered the most significant architectural development in the last 5-10 years. One hundred man-years were spent on the silicon design and layout alone. The design goals of iAPX 432 are the following:

(*i*) The principal objective of the 432 is to significantly reduce the cost and time to develop software-intensive microprocessor applications.

(*ii*) A second objective of the 432 system is incremental performance, or the ability to tune the performance of the system by adding or subtracting processors without the need to change one's software.

(*iii*) A third objective is reliability. This included the ability for programs to handle both hardware and software detected fault conditions.

(*iv*) A fourth objective is support for the Ada language.

In many respects, the Intel iAPX 432 is very different from the chips described above. The others are single-chip CPUs, but the iAPX 432 is a three-chip set. The fundamental concept of the architecture is objects. An object is a collection of related information which, with a set of applicable operations (largely machine instructions), form an abstruction. The General Data Processor system, or GDP, is a

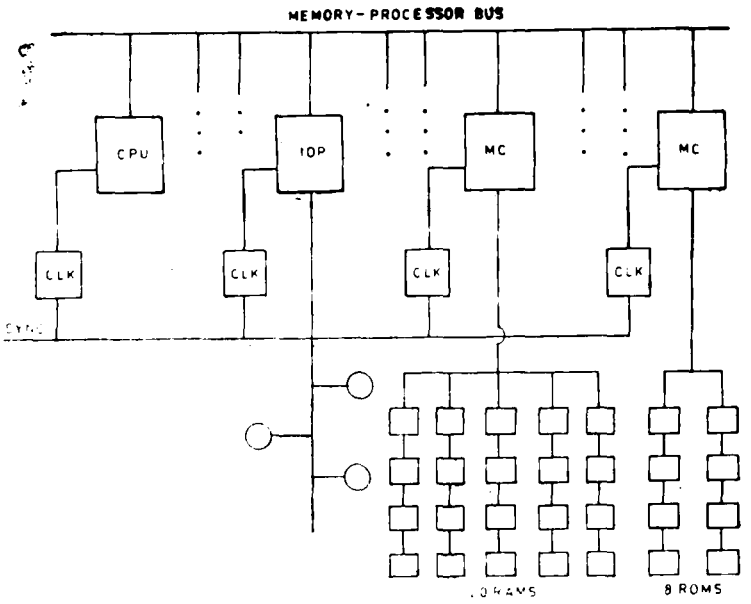

Fig 4 HP 32-Bit microcomputer system

subset of the iAPX 432. It consists of two chips: the iAPX 43201, with 110,000 transistors which is responsible for instruction decoding and the iAPX 43202, with 49,000 transistors, which performs actual instruction execution. The iAPX 43203 I/O interface processor, containing 60,000 transistors, performs I/O interface functions and possesses limited capabilities for execution of microcode. There is an additional chip—a multiprocessor memory interface—that provides the signals required for packet bus interconnection.

The operation of the iAPX 432 involves communication among the iAPX 43201 instruction fetch and decode unit, the iAPX 43202 instruction execution unit, and the iAPX 43203 I/O interface unit. This communication is handled over the processor-memory interconnect bus, as shown in Fig 5. The characteristics of the iAPX 432 chips are summarized in Table 2. Data can be manipulated in the form of eight-bit characters, 16/32 bit ordinals, 16/32-bit integers, 32/64/80-bit floating-point variables, bit strings, arrays, records, or objects, which are data structures containing information organized in some manner. Such objects can be referenced as a single entity, their internal organization is hidden and protected from all other procedures by hardware mechanisms. Each object has defined for it a set of operations (procedures or instructions) that are permitted to directly manipulate it. Examples of hardware defined objects are:

* processor objects, which represent the physical processors;
* process objects, which represent the individual computing tasks;
* Context objects, which represent the activation of a program unit;
* despatching-port objects which provide a stream of work for a set of processors; and
* communication-port objects, which support interprocess communication and synchronization.

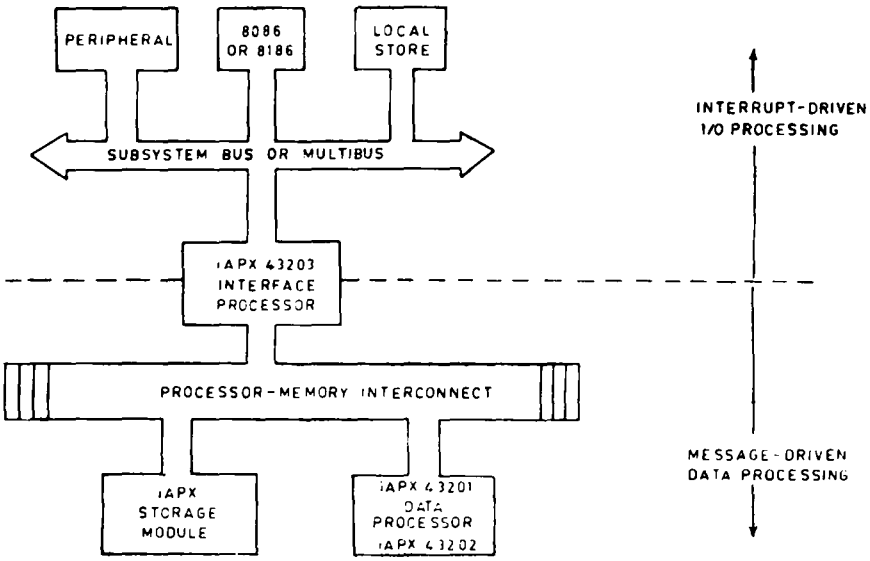

Fig 5 Intel iAPX 432 three-chip 32-bit microprocessor

Table 2 [7] *Characteristics of the components of the iAPX 432*

Characteristic	*iAPX 43201*		*iAPX 43202*		*iAPX 43203*	
Die Size (in m)	318x323		366x313		358x326	
Total device placements	110,000		49,000		60,000	
Function of unit	Instruction fetch and Decode		Instruction Execution		I/O Interface	
Functional Sub-units	Instruction Decoder	Microinstruction sequencer (MIS)	Data Manipulation unit (DMU)	Reference generation unit (RGU)	Data Acquisition unit (DAU)	Micro-Execution unit (MEU)
Function of Sub-unit	Decodes Variable-length, Bit-aligned Instructions	Sequences Vertically encoded Micro-Instructions and inputs them to the MEU	Contains operand and units to implement the macro-instruction Set Efficiently	Contains registers and Functional units for Logical-to-Physical address Translation and access Right Verification	Performs Perefetch and Post write Buffering of data and Generates Main system Momory Accesses	Performs system Access, environment Manipulation, interprocessor communication and Address Map Set-up

The general data processor (GDP) is a 32-bit VLSI processing unit that serves as the primary computational engine for the Intel iAPX 432 micromainframe. The GDP consists of two VLSI chips, the 43201 instruction decoding unit (IDU) and the 43202 microinstruction execution unit (MEU). The GDP is implemented as a three stage pipe line. The first stage of the pipe line is the instruction decoder (ID) which receives the instruction stream data and produces the logical addresses and microinstruction necessary to execute the instruction. The second stage of the pipeline, the microinstruction sequencer (MIS) controls the flow of microinstructions and data to the microinstruction execution unit (MEU). The last stage of the pipeline is the microinstruction execution unit which actually performs the sequence of microinstructions necessary to execute the instruction.

The basic function of the MEU is to accept decode and execute any of the 88 unique vertically encoded microinstruction supplied by the IDU. These microinstructions control the regsiters and the arithmetic capabilities of MEU to perform the GDP macroinstruction. The MEU is also responsible for interfacing with main memory; thus, it supports the required virtual-to-physical address translation mechanism and initiate all memory accesses on behalf of the processor.

The implementation of the GDP departs from conventional processor implementation in several ways. No processor interrupts or service requests are recognized by the GDP. It does not support the concept of input/output address space. Instead, all input/output is accomplished using a hardware-supported message-based communication mechanism which is referred to as an interprocess communication. Thus messages are sent through an interface processor, using objects (data structures) in the main memory as the message buffering medium, to an attached I/O processor that performs the actual I/O operations with its associated I/O subsystem.

The 43203 interface processor (IP) serves as a data and control interface between the protected 432 central system and an I/O subsystem. A subsystem processor called the attached processor (AP) provides the intelligence to control I/O activity in the subsystem and transfers data through the IP between the subsystem and the 432 memory space. The IP acts as a slave to the AP and maps a portion of the AP's address space, called a window, into a segment of the 432 system memory known as an object. The IP operates in conjunction with the AP to logically form a 432 I/O processor.

There are no programmer-visible registers defined by the architecture. A microinstruction execution unit performs several functions traditionally associated with registers. The MEU is comprised of two semi-autonomous subprocessors, a data manipulation unit (DMU) and a reference generation unit (RGU). The DMU contains the operand registers and functional units necessary to efficiently implement the macroinstruction set. The RGU contains the registers and the functional units that support logical-to-physical address translation and access rights verification.

Instruction codes for the GDP are designed to minimize the memory space occupied by the instruction while still allowing efficient encoding instructions without regard to byte, word, or other artificial boundary. The instruction streams thus consist of a linear string of bits, with each instruction occupying exactly the number of bits it requires. No extra bits are appended to bring instructions to a standard length.

All 432 machine instructions have the format shown in Fig 6. The four parts of an instruction are the class field, which specifies the number of operands and their lengths, the format field, which specifies the order of the operands and whether they are represented by data references or found on the stack, data/-branch references, and the operation code, which specifies the operation to be performed. The class field and operation code are not independent; in other architecture both of these are combined into what is known as the operation code. The largest possible instruction is 321 bits; the smallest instruction is 6 bits.

Class	Format	Data/branch references	Operation code
4-6 bits	0-4 bits	0-306 bits	0-5 bits

increasing address

Fig 6 432 Instruction format

Like the HP processor, the iAPX 432 performs run-time bounds checking of addresses. The iAPX 432 supports integer data in the form of half-words and words and floating-point data in the form of words, double-words, and as 80-bit quantities. The total number of instructions is 230. By allowing both stack and memory-to-memory arithmetic, the iAPX 432 can provide denser code and faster speeds

for evaluating expressions. The instruction set supports objects through messages (SEND, WAIT), context (CALL, RETURN), storage pools (ALLOCATE, TYPE), and processes (SCHEDULE, DESPATCH).

There are at least seven levels of indirection between the instruction and its operand because one must pass through the object table directory twice. The overhead implied by this indirection is usually not present for two reasons. One is that local variables, which should be most frequently referenced data items, are usually located in a data segment associated with the context object. Second, each processor maintains several on-chip associative memories.

The 432 provides a large address space, in terms of both objects and physical storage. The encoding of capabilities (access descriptors) provides a maximum range of 2^{24} objects (more accurately 2^{24} segments). A segment can have size of 65,536 bytes. Thus the maximum amount of addressable virtual space is 2^{40} bytes. A two-step mapping process separates the relocation mechanism from the access control mechanism. Segments are of two types—access and data. The hardware recognizes them differently and rigorously enforces the distinction. The iAPX 432 provides four addressing modes : the base and index direct, used to access scalars; the base indirect, index direct, used to access records; the base direct, index indirect, used to access static arrays; and the base and index indirect, used to access dynamic arrays. The emphasis is on addressing objects through the use of access descriptors in the form of directory index and segment index.

According to Justin Rattner, Intel's Chief architect for the 432, the following features of the 432 complement the Ada language and increase programmer productivity.

(i) Memory management : The 432 provides machine instructions in hardware that correspond directly to statements in Ada dealing with the allocation and deallocation of memory; whereas, in a conventional machine, software instructions would have to be invoked in an Ada run-time system.

(ii) Garbage Collection : Described as one of the principal sources of errors in sophisticated programs, garbage collection is the process of discovering which areas of memory are no longer being used. This is particularly important in that when a process runs out of memory, it simply stops. Or, at least, program execution stops while the system proceeds to scan memory, tooking for those portions that are no longer in use.

In Ada, storage can be allocated in extremely flexible ways, but the language does not require the programmer to notify the machine or the operating system when he is no longer using that storage. The advantage is that the programmer is no longer required to manage his storage explicitly. Instead, the garbage collection task is left either to the hardware or to the operating system software, thus freeing the applications programmer from having to constantly keep track of memory, one of the most demanding of intellectual activities.

In the 432, the hardware does that part of the garbage collection process called marking, and the software the other part, known as reclamation. Since the 432 is also multiprocessor, a second processor can be used to run the reclamation activity while other Ada programs are executing on other processors.

(iii) Object level protection: Particularly important for command and control systems that require a very high level of information security, both Ada and the 432 incorporate methods of protecting data on a 'need to know' basis.

While Ada allows programmers to define data structures and protect them from misuse or accidental damge, the 432's addressing mechanism provides an additional level of security. In the 432, only the hardware is capable of creating an address. Every address, therefore, becomes a protected entity, and it is impossible to forge an address for any data structure. Once the address has been created, it cannot be modified by a random program. When a program is executing, access to information is limited strictly to those program modules containing the names of the data structures where the information is stored.

The same concept of program modules in Ada extends 'need to know' security to the program coding environment as well. A programmer need only know the names, not the details, of procedures or data structures used in other modules, and then only if they are applicable to his individual assignment.

(iv) User defined data types : Data typing and the representation of programmer defined data types is a technique enabling the programmer to turn the 432 from a general purpose machine into an applications specific machine by defining those data structures necessary for this application. Ada is a strongly typed language, and the 432 provides all of the protection, checking and access controls need to fully support this concept.

(v) Program Modularization : In a conventional computer, the structure of a program is often lost when it is compiled, becoming just a string of bits. In an Ada program written for the 432, this structure is preserved, enabling the machine to report errors in the program and relate them to specific data structures. The 432's addressing mechanisms allow it to recognize the programmer defined data structures and say, 'you've damaged this structure', or you have done something, or attempted to do something, improperly within structure XYZ.

(vi) Multiprocessing : Ada is one of the few languages and perhaps the first standard language, to incorporate facilities for multiprocessing. The initial 432 system offered supports upto six processors and later 432 systems will support at least twice that number. One of the primary advantages of the 432 may be the ability to add processors without having to change any software, providing a graceful way to increase processing power incrementally as system requirements evolve over time.

(vii) Interprocess Communication : Known in Ada as task communication or the rendezvous concept, interprocess communication buffering and synchronization facilities built into the 432 allow the machine to automatically transmit messages from one task to another.

REMARKS

All the chips reflect a conscious attempt towards the integration of an enormously large number of transistors. Even with a significantly smaller chip size, Hewlett-Packard has packaged the largest number (450,000) on a single chip, almost twice what Intel has packaged on three chips put together. The Intel iAPX 432 is also implemented in NMOS. Because of the CMOS implementation, the Bellmac-32A chip is the least power-consuming microprocessor.

The timing estimates for several elementary operations are summarized in Table 3. Actual throughput is a function of the exact instruction sequence, displacements, data lengths, clock frequency and other factors. Overall, the performance of the newer microprocessors

Table 3[1] *Capabilities of the four microprocessors, not including functions provided by coprocessors or auxiliary chips*

	NS 16032	*BELLMAC-32A*	*HP32*	*iAPX 432*
SYSTEM STRUCTURES				
Uniform addressability	△	△	△	△
Module map and modules	△	×	×	△
Virtual	△	△	△	△
PRIMITIVE DATA TYPES				
Bits	△	△	△	△
Integer byte or half-word	△	△	△	△
Integer word	△	△	△	△
Logical byte or half-word	△	△	△	△
Character strings (variable)	△	△	△	△
BCD byte or half-word	△	×	×	×
BCD word	△	×	△	×
32-bit floating-point	×	×	△	△
64-bit floating-point	×	×	△	△
80-bit floating-point	×	×	×	△
DATA STRUCTURES				
Stacks	△	△	△	△
Arrays	△	△	△	△
Packed arrays	△	×	×	×
Records	△	×	×	△
Packed records	△	×	×	×
Strings	△	△	△	△
PRIMITIVE CONTROL OPERATIONS				
Condition code primitives	△	△	△	△
Jump	△	△	△	△
Conditional branch	△	△	△	△
Iterative loop control	△	×	△	△
Subroutine call	△	△	△	△
Multiway branch	△	△	△	△
Orthogonal instruction set	△	△	△	△
CONTROL STRUCTURE				
External procedure call	△	△	△	△
Semaphores	△	△	△	△
Traps	△	△	△	△
Traps	△	△	△	△
Interrupts	△	△	△	△
Supervisors call	△	△	△	△
Objects	×	×	×	△
Hierarchical operating system	×	△	×	△
OTHER				
User microcode	×	×	×	×
Debug mode	×	△	△	×
Compatibility with other microprocessors	×	×	×	×
Self-test during power-up	×	×	△	×

△—Feature available
×—Feature not available

approaches the performance of mainframes. In terms of basic computational power, the iAPX 432 is superior to an IBM 370/148, and the Bellmac-32A and the Hewlett-Packard processors are expected to be superior compared to an IBM 370/158. However, whereas the IBM 370 family is supported by extensive software in terms of compilers and application programs, it would take some time for similar facilities to be available on 32-bit microprocessors.

Overall performance is determined by the hardware and by the efficiency of the several layers of software that determine how a user uses his chip. The trend in microprocessors is toward microcoding and performing traditionally software-based functions in hardware. For example, the operating system function of job despatching is performed entirely by hardware in the iAPx 432. The hardware shift circuit is another example. Today one must analyze the match between the user's application and the functions being implemented in hardware.

Concurrent with the development of basic processors, manufacturers have developed sophisticated chips for auxiliary functions—memory management, control of DMA operations, control of peripherals, bus management and arbitration control of communications, and control of input and output functions. However, a particular chip of this kind is designed to support only a particular family of microprocessor chips of a particulars vendor. Usually there is a time lag between the introduction of the processor chip itself and the introduction of support chips.

To increase computational bandwidth and/or system resilience, integration of several microprocessors in a single system frequently becomes necessary. The overall throughput and efficiency of such system is directly dependent on the hardware and software interconnection mechanisms supported by the basic microprocessor chips.

For the growing class of 32-bit microprocessors, developments favoured implementation in CMOS, integration of support functions such as cache memory and memory management on the chip. The MC68020 represents the successful extension of a 16-bit microprocessors into the 32-bit world. It provides a full 32-bit data and address bus as well as a 32-bit architecture. It is designed to execute all user object code written for previous members of the M68000 family of processors and to execute these programs much faster. Because of the value of the large software base available for M68000 processors, the MC68020 maintains compatibility with previous members of the family. In addition to performance and compatibility, it supports additional addressing modes, includes new instructions, extends all instructions to 32-bit operations, has a larger register set, and supports more data types. The MC68020 supports the virtual memory/virtual machine functionality of the MC68010, extends the 68010 support for modular programming, and provides a coprocessor interface for instruction set extension.

Although all MC68000 family members have 32-bit user registers, operands and internal registers, the MC68020 adds full 32-bit data paths (internal and external), two 32-bit internal address paths, a 32 bit execution unit three 32-bit arithmetic units, and an on-board instruction cache. The design goal was to provide a four-to sixfold performance improvement for the 16-MHz MC68020 over the 8-MHz MC68000. This performance increase is realized by a combination of the higher clock frequency at which the MC68020 operates, the 32-bit data bus width, the presence of the on-chip instruction cache, and the addressing mode enhancements.

The NS32C032 will be CMOS version of National's NS32032. The NS32332 will be software- and hardware-compatible with the NS32032, but will execute instructions to-and-a-half to three times faster than NS32032. The NS32132 will be a version of the NS32032 modified for multiprocessing applications.

ATT's new high-preformance 32-bit microprocessor is the WE32100 and its predecessor CPU chips are the Bellmac-32, 32 A', and 32A. This second-generation version CPU (of the 32A) has fully demultiplexed 32-bit address and data buses, 108 active pins, and 180,000 transistors in an area of 1 sq. cm. (150,000 sq. mil), a 10-MHz internal machine cycle, and a power dissipation of less than one watt. According to ATT, the 32-bit WE32100 CMOS microprocessor represents a three-fold increase in performance over its predecessor the WE32000.

Intel Corp. continues to work on the 80386. It will be software compatible with the other members of the '86' family.

Improvements in VLSI technology enabled microprocessors to offer increased functional capabilities at diminishing costs over the years. Enhancements in the areas of processing power, peripheral support, and in terms of software are likely to continue. The field of microprocessors is dynamic and one can expect newer and more powerful chips.

REFERENCES

1. J.A. Bayliss *et al.*, The Interface processor for the Intel 432 32-bit computer, *IEEE Journal of Solid State circuits*, vol SC-16, pp 522-530, Oct 1981.
2. J.A. Bayliss *et al.*, The Instruction decoding unit for the VLSI 432 General Data Processor, *ibid.*, vol SC-16, pp 531-536, Oct 1981.
3. J.W. Beyers *et al.*, A 32-bit VLSI CPU chip, *ibid.*, vol SC-16, pp 542-547, Oct 1981.
4. D.L. Budde *et al.*, The execution unit for the VLSI 432 general data processor, *ibid.*, vol SC-16, pp 514-521, Oct 1981.
5. G.J. Myers, *Advances in computer architecture*, John Wiley and Sons, 1982.
6. R. Govinda Rajulu, A review of 16-bit microprocessors, *Students' Journal of IETE*, vol 24, pp 143-150, Oct 1983.
7. A Gupta & H.D. Toong, An architectural comparison of 32-bit Microprocessors, *IEEE Micro*, vol 3, pp 9-22 Feb 1983.
8. A. Gupta & H.D. Toong, Microprocessors—The first twelve years, *Proc IEEE*, vol 71, pp 1236-1256, Nov 1983.
9. A. Kaminker *et al*, A 32-bit microprocessor with virtual memory support, *IEEE Journal of Solid-State Circuits*, vol SC-16, pp 548-557, Oct 1981.
10. J. Rattner & W.W. Lattin, Ada determines architecture of 32-bit microprocessor, *Electronics*, pp 119-126, Feb 1981.
11. Subhash Bal *et al.*, The NS16000 Family—Advances in Architecture and Hardware, *IEEE Computer*, vol 15, pp 58-67, Jun 1982.
12. D. MacGreger, D Methersole and B. Moyer, The Motorola MC68020, *IEEE Micro*, pp 101-118, Aug 1984.
13. J.C. Patterson, Extending Ada into silicon, *Defence Electronics*, vol 13, Sept 1981.