Z80,000 32-bit microprocessor

by ANIL PATEL*
Zilog, Inc.
Campbell, California

ABSTRACT

With recent advances in very large scale integrated (VLSI) circuit design, the formerly distinct boundaries between micro-, mini-, and mainframe computer architectures are eroding. For example, the Motorola 68000, the Z8000 CPU, and the Intel 8086 have broken the boundary between micro- and minicomputer. Now, the Z80,000 CPU, Zilog's new 32-bit processor chip, has broken the distinct boundary between mini- and mainframe computers by featuring a mainframe power on an integrated-circuit chip. The distinguishing features of the Z80,000 CPU—such as on-chip virtual memory management, on-chip cache memory, six-stage pipeline architecture, burst memory transfer, and multiprocessing support, put it ahead of any CPU in its class. It supports linear and segmented addressing. The regular instruction set and rich and powerful addressing modes are well suited to compilers and operating systems. The flexibility and simplicity of the Z80,000 provide a simple solution to hardware and software system design.

* Anil Patel is not currently at Zilog, Inc.
OVERVIEW

The Z80,000 32-bit microprocessor integrates major system functions on a single chip while also providing extensive system design flexibility. The microprocessor offers (1) Full 32-bit address and data paths; (2) on-chip instruction and data cache; (3) on-chip, demand-paged virtual memory management; (4) an execution rate of 1 to 12.5 million instructions per second (MIPS), and (5) burst transfer capability, providing enhanced bus bandwidth. A six-stage pipeline is incorporated to enhance throughput and to maximize the effectiveness of the on-board cache and memory management unit. Additional features include (1) The largest register file of any microprocessor (16 32-bit general-purpose registers), (2) nine general addressing modes, (3) a powerful instruction set directed toward support of high-level language programming, and (4) memory mapped I/O.

The Z80,000 CPU is an upward compatible extension of the Z8000 family of microprocessors and can be used with the complete line of Z8000 peripheral and software support.

ARCHITECTURE

The Z80,000 uses 32-bit logical addresses to access directly four billion bytes of memory in each of four address spaces. Separate address spaces are provided for system and normal modes and for instructions and data. Programmers have a choice of three different address representations in accessing memory space (Figure 1). Two bits in the flag and control word select compact, segmented, or linear addressing modes.

Compact mode uses a 16-bit address and is compatible with the Z8000 nonsegmented mode. Programs using less than 64k bytes of address space can take advantage of the compact mode's dense code and efficient use of base registers.

For larger programs, many applications prefer to use segmented mode, in which individual objects are allocated to separate protected segments. During address calculation the segment number is unchanged and only the offset is affected. The Z80,000 provides two segment sizes, which allow the flexibility of having both 128 segments of 16 Mbyte maximum size, and 32,728 segments of 64 Kbyte maximum size. The most significant bit of the logical address selects between the two segment sizes.

The third mode supported by the Z80,000 is 32-bit linear addressing. Applications requiring the flexibility of a large linear address space without the formal structure of segmentation can use the linear addressing mode.

A separate system mode, with its own stack pointer and protected address spaces, supports operating systems. Further, some instructions are privileged and execute only in system mode. Thus, systems software and resources are protected from normal mode programs, which can use the System Call instruction and traps to request operating system services. Operating systems also benefit from instructions aiding system call processing, context switching, and resource allocation and process synchronization.

The instruction set of the Z80,000 CPU is oriented toward compilation of high-level-language programs into compact, efficient code. The register file includes sixteen 32-bit general-purpose registers, which can hold data accumulations, index values, and memory addresses. Nine general addressing modes provide efficient access to the many types of data structures commonly used. Further, these can be combined with operations on a variety of data types (including 8-, 16-, and 32-bit integer and logical values, bits, bit fields, packed BCD bytes, and dynamic length strings) to provide a powerful instruction set. Additionally, compilers are aided by instructions for procedure linkage, array indexing, integer conversions, and other common operations.

The Z80,000 also includes extensive trap facilities for runtime error detection and software debugging, thus enhancing system development and reliability.

MEMORY MANAGEMENT

Features offered by the Z80,000's paged memory management unit include:

1. Translation of logical addresses to physical addresses
2. Memory access protection
3. A translation-lookaside buffer (TLB)
4. Protected access to memory-mapped I/O devices

![Figure 1—Address representations](From the collection of the Computer History Museum (www.computerhistory.org))
Demand-paged virtual memory is easily implemented without special software fixup routines or storage of internal state following address translation faults. The implementation is accomplished through early detection of translation faults resulting in the ability to restart all instructions efficiently.

The Z80,000 implements a three-level address translation process. Once the operating system creates the translation tables in memory and initializes the on-chip translation table descriptor registers, the Z80,000 automatically references the Level 1, Level 2, and page tables to perform address translation and access protection (Figure 2).

Access protection information is encoded in a four-bit protection field at any level of the translation process. The protection field is defined for all translation table descriptors and translation table entries, but only one protection field is selected during an address translation. This allows access protection to be done at the page level, segment level, or even a mixture of these two approaches.

It is possible to reduce the number of levels of translation required by specifying in the table descriptor register that either or both Level 1 and Level 2 tables should be omitted during the translation process. Skipping the Level 1 tables is useful only when the logical address space is needed; skipping the Level 2 tables can be used for Z8000-compatible programs; skipping both Level 1 and Level 2 tables is useful for compact mode programs. When a level is skipped during address translation, the corresponding bits in the logical address are ignored. In addition, the size of the tables can be reduced by specifying, in the table entries, the size of the next level tables in increments of 256 bytes; or by specifying that the entries and the tables they point to are to be considered invalid.

Delays associated with referencing translation tables are minimized by using a special on-chip buffer called the translation lookaside buffer (TLB). Logical addresses and their corresponding physical addresses are stored in the TLB as they are translated through the translation tables. Subsequent accesses to the same page do not require translation through the memory resident translation tables, but are simply retrieved from the TLB (Figure 3). A translation through the tables is performed only when an entry does not exist in the TLB (a TLB miss). The least recently used entry is then replaced with the new address and its translation. The TLB can hold the 16 most recently referenced pages.

Bit 31 of the translated physical address selects either memory-mapped I/O space or memory space. If translation is enabled, any reference to I/O space carries I/O status and timing for the reference. The address translation process therefore allows normal mode (user) programs direct protected access to I/O devices without operating system overhead.

**CACHE**

The six-stage pipelined architecture of the Z80,000 incorporates an on-chip cache to buffer the high execution rate of the pipeline from relatively slow memory access rates. The pipeline may require two memory fetches during each processor cycle, one for the instruction fetch stage, and one for the operand fetch stage. Since main memory fetches may normally require two processor cycles, the pipeline would be idle if all references had to access main memory. The on-chip cache, though, can be accessed every clock cycle, thus satisfying most memory references without external bus transactions.

The cache effectively decouples the internal execution rate of the Z80,000 from the external bus transfer rate. Hence the system designer is able to make different memory subsystem cost-performance tradeoffs independent of the CPU. The Z80,000 bus interface can then be configured to match the CPU's clock rate to the rest of the system. This also enables easy incorporation of future higher performance versions of the Z80,000.

Copies of the most recently referenced memory locations are stored in cache. During a memory access, the address of the desired memory location is presented to the cache tag memory and if there is a match (a cache hit), then the data are read from the cache and no external memory transaction is performed. The hit ratio is typically 90% for instructions and 75% for operand fetches. If there is not a match (a cache miss), the CPU generates an external bus transaction to fetch the data from main memory, and the data are placed in the cache. The pipeline stage requesting the missing data waits until the external transaction is complete; the other stages can continue to operate.

The cache stores 16 blocks of information from main memory. Each block contains 16 bytes, providing 256 bytes of storage. Associated with each block of cache data is a 28-bit tag field. This tag field contains the 28 most significant bits of
address corresponding to the block. The lower four bits of an address select the desired byte, word, or longword within the block. Each word within a block also has an associated Valid bit indicating whether the cache is holding a valid copy of the corresponding memory location (Figure 4).

When a cache miss occurs, a least recently used (LRU) algorithm is used to select a cache block to be replaced. If an instruction fetch causes a cache miss, the CPU fetches the missing instruction from memory. In addition, if the memory supports burst transactions, the balance of the block is prefetched. When a cache miss occurs on an operand fetch, then the CPU fetches only the missing data from memory. Missing operands are selectively prefetched only for instructions for which further operand fetches are anticipated.

For references which require operand stores, the data are always written to main memory. The cache is also updated if it contains the addressed location; otherwise it is unaffected. This mechanism, called writethrough, ensures that main memory always holds up-to-date data. In systems where multiple processors may share writable storage, the address translation tables allow individual pages to be marked non-cacheable.

Unlike some processors, which cache only instructions, the caching mechanism of the Z80,000 can selectively cache instruction fetches and/or operand fetches. While particular applications for the Z80,000 may choose to cache only instructions, caching data along with instructions typically improves cache performance by 20%.

BUS STRUCTURE

To meet particular cost and performance objectives, the system designer can choose the memory data width, access time, and bus bandwidth simply by setting the appropriate bits in the hardware interface control register (HICR). Increased bus bandwidth can be achieved by taking advantage of the optional burst transfer capability of the Z80,000 bus interface. The Z80,000 also provides support for different types of multiprocessor configurations.

The HICR specifies certain characteristics of the hardware configuration surrounding the CPU, including bus speed, memory data path width, and number of wait states automatically inserted. The bus speed is chosen to be either ½ or ¼ of the clock frequency. The memory data path width can be specified separately for the upper and lower portions of memory space (M0 and M1) as 16 or 32 bits. The number of Wait states automatically inserted by the CPU for references to the different memory and I/O spaces (M0, M1, I/O0, I/O1) may also be separately specified. Thus, a system can easily accommodate a slow, 16-bit-wide bootstrap ROM in one region and a fast, 32-bit-wide RAM in another region.

Burst memory transactions use multiple Data Strobes following a single Address Strobe to transfer data at consecutive memory addresses. The transactions are controlled by the BRST and BRSTA signals, which allow either the CPU or memory to terminate a burst transaction after any number of transfers. The CPU’s protocol supports the use of interleaved memory systems and allows the system designer to easily take advantage of nibble-mode RAMs. The CPU uses Burst transactions to prefetch a cache block on an instruction fetch cache miss. They are also used to fetch or store operands when multiple transfers are necessary, as with unaligned operands, string instructions, Load Multiple instructions, and loading of program status.

The CPU provides support for interconnection in four types of multiprocessor configurations (Figure 5):

1. Coprocessor
2. Slave processor
3. Tightly-coupled multiple CPUs
4. Loosely-coupled multiple CPUs

Coprocessors work concurrently with the CPU to execute a single instruction stream, using the Extended Processing Architecture facility. This allows extension of the Z80,000 architecture to include floating point operations and other specialized functions. Additionally, the processing speeds offered by extended processing units (EPUs) optimized for particular operations, such as the Z8070 Arithmetic Processing Unit, can provide significant performance improvements. The signal EPUBSY is dedicated as a synchronization signal for connection with coprocessors. In systems supporting the functionality of an extended processing unit without the actual EPU present, the EPU instructions are trapped and emulated in software.

Slave processors, such as the Z8016 Data Transfer Controller, perform DMA functions asynchronously to the CPU. The CPU and slave processor share a local bus, of which the CPU
is the default master, using the CPU's BUSREQ and BUSACK lines.

Tightly coupled multiple CPUs execute independent instruction streams and generally communicate through shared memory located on a common (global) bus using the CPU's GREQ and GACK lines. Each CPU is default master only of its local bus, while an external arbiter chooses the global bus master. The CPU also provides status information about interlocked memory references so that bus control is not relinquished during an indivisible operation such as Test and Set or Increment Interlocked.

The Z80,000's I/O and interrupt facilities support loosely coupled multiple CPUs, which generally communicate through a multiported peripheral, such as the Z8038 Z-FIFO I/O Controller.

PIPELINED ARCHITECTURE

The Z80,000 has a six-stage pipelined architecture. The following functions are performed at each of the stages:

1. Instruction fetch
2. Instruction decode
3. Address calculation
4. Operand fetch
5. Instruction execution
6. Operand store

For simpler instructions all stages of the pipeline (except for operand store) are completed in one processor cycle (two clock cycles) (Figure 6). Therefore at 25 MHz the peak instruction execution rate is 12.5 MIPS. In practice the actual instruction execution rate is about one-third the peak rate, depending on the instruction mix and system configuration.

The on-chip cache is time-multiplexed between instructions and data, allowing for the occurrence of instruction and data accesses within the same processor cycle. The pipeline then flows smoothly without having two stages competing for the same resource. Similarly, there are two ALUs, one for address calculation and one for instruction execution. This eliminates sharing a single ALU between the address calculation stage and the execution stage. The on-chip TLB also helps smooth the pipeline flow, as the address translation is done in the address calculation stage without having to go off-chip. The operand storage stage is the only stage requiring several cycles. Performance is not seriously degraded, however, because the store operation can usually overlap with the processing of other instructions.

Z80,000 CPU PERFORMANCE

Cache memory and the pipelined structure cause the performance evaluation of the Z80,000 CPU to be complex. The best approach is separation of instruction processing time into a sum of three components: execution time, pipeline delays, and memory delays. Performance was evaluated by statistically measuring activities of ten C language programs, and then performing a computer simulation of the cache, translation-lookaside buffer, and pipeline mechanisms.

The execution time for an instruction is the number of cycles required to execute the instruction if no other delays, such as a cache miss or register interlock, occur. Common instructions, such as loading a register with a word operand specified by a base-register-plus-displacement addressing mode, execute in one processor cycle (two clock cycles), but the average instruction execution time is 1.3 processor cycles.

Pipeline delays are caused by branch instructions, register interlocks, and other miscellaneous delays. The most significant of these delays is the delay due to branch instructions. When a branch is taken, instructions in the pipe behind the branch instruction are flushed. Unconditional branches introduce a delay of two processor cycles. Conditional branches cause a three-processor-cycles delay if the condition is met, and no delay if the condition is not met. The average delay due to branches is 0.5 processor cycle per instruction.

Another significant pipeline delay is register interlock. Whenever the execution stage modifies a register to be used in a subsequent instruction as an address register, the address calculation must be held up (interlocked) until the execution is complete. The interlock ensures that the proper register value is used in the address calculation. A register interlock occurs for 11% of instructions, causing a 0.19 processor cycle delay, and a cache reference interlock occurs for 6% of instructions, causing a 0.11 processor cycle delay. Therefore the total average pipeline delay is 0.9 processor cycle per instruction.

Memory delays are caused by cache misses and TLB misses. When the processor fetches an instruction or operand for which a corresponding entry in the cache or TLB does not exist, a reference to main memory is generated. The average delay due to these memory transfers is calculated assuming a 32-bit data path, a cycle time of two processor cycles for read transactions, a cycle time of three processor cycles for write transactions, and support of burst transfers. An average delay per instruction due to cache misses is 0.4 processor cycle, and an average delay per instruction due to TLB misses is 0.5 processor cycle. Therefore, the total memory delay with a Memory Management Unit is 0.9 processor cycle. The memory delay without memory management is 0.4 processor cycle.
Instruction processing time $T_i = \text{Execution delay} + \text{Pipeline delay} + \text{Memory delay}$. Therefore, $T_i = 1.3 + 0.8 + 0.9 = 3.0$ processor cycles. Ti without MMO = $1.3 + 0.8 + 0.4 = 2.5$ processor cycles. At 25 MHz, this corresponds to 4.2 MIPS. At 25 MHz, without MMO, the CPU’s average sustained performance is 5 MIPS.

The above analysis assumes that these delays occur in sequence. In reality simultaneous delays cancel each other, and consequently the actual CPU performance may be better than the calculated CPU performance.