The 80386SX Processor Bus

This lecture describes the signals and operation of the Intel 80386SX processor bus.
After this lecture you should be able to draw timing diagrams for the processor bus signals described in this lecture and state the values that would appear on the data and address buses. You should be able to do this for memory and I/O read and write cycles and for interrupt acknowledge cycles.

History

Intel's first 16-bit CPU was the 8086. A version of the 8086 that used an 8-bit data bus, the 8088, was released later to permit lower-cost designs. The 8088 was used in the very popular IBM PC and many later compatible machines.

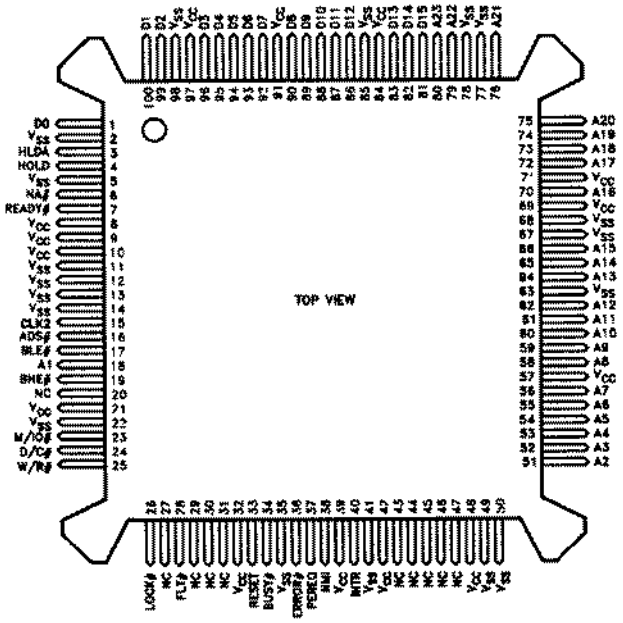
Intel's first 32-bit CPU was the 80386. It was designed to be backwards-compatible with the large amount of software which was available for the 8086. The 80386 extended the data and address registers to 32 bits. The Intel '386 also included a sophisticated memory management architecture that allowed *virtual memory* and *memory protection* to be implemented. This same basic 80386 architecture is used present in the latest generation of Pentium and compatible processors.

In this lecture we will describe the processor bus of the Intel 386SX, a version of the 386 with a 16-bit processor bus. The 386EX, the chip that we will use in the lab, is similar to the 386SX but also integrates several commonly-used peripherals in the same chip.

Processor	Register	Data Bus
8086	16	16
8088	16	8
i386	32	32
i386SX	32	16
i386EX	32	16

'386SX CPU Signals

The 386SX is packaged in a 100-pin package. It has a 24-bit address bus and a 16-bit data bus. The names of the signals are shown below:



Utility Bus

The utility bus includes the pins that are required for the processor to operate properly but are not involved in data transfers. This includes the power, ground, and clock pins.

The two most important pins on the CPU chip are for power supply and ground. The processor will operate erratically (or not at all) unless the power supply is held at the proper voltage.

The next most important signal is the clock, CLK2. Output signal transitions happen on the rising edge of CLK2 and inputs are also sampled on the rising edge of CLK2.

Figure 1 shows examples of data transfers over the processor bus. Each transfer (read or write) is called a *bus cycle*. Each bus cycle requires two or more *processor cycles* (one T1 cycle plus one or more T2 cycles). Each of these processor cycles requires two

CLK2 periods. Figure 1[1] shows how two CLK2 cycles make up a processor cycle and how two processor cycles (T1 and T2) make up a bus cycles.

Cycle		Requires
bus cycle	$>= 2$	processor cycles
processor cycle	2	CLK2 cycles

Exercise: A 386SX CPU is operating with a 25 MHz CLK2 signal. What is the CLK2 period? How long does a processor cycle take? How long does a bus cycle take?

Address and Data Busses

The 80386SX has a 16-bit data bus and a 24-bit address bus. These signals are labelled D_{15} to D_0 and the A_{23} to A_1 (not A_0) respectively. To allow for either 8-bit or 16-bit transfers the chip uses BHE* and BLE* (high- and low-byte enable) signals indicate to memory and I/O devices which byte(s) being transferred. The BHE* indicates a transfer over D_{15} to D_8 and BLE* indicates a transfer over D_7 to D_0. BHE* and BLE* also indicate the memory address being accessed: BLE* and BHE* indicate addresses with $A_0 = 0$ and $A_0 = 1$ respectively.

Unlike the Motorola 68000, the intel chips allow 16-bit values to be written to odd addresses and 32-bit values to be written to addresses that are not multiples of 4 (i.e. memory operations do *not* have be *word-aligned*). Thus the value transfered over the high-order byte of the data bus may not correspond to the high-order byte of the value being written.

Endianness

Intel processors, unlike Motorola processors, use so-called "little-endian" byte order. This means that 16- or 32-bit words are stored with the *least-significant* byte at the lowest-numbered address. This can be confusing. We normally write memory contents in increasing address order from left to right; in little-endian storage order the bytes in multi-byte words appear in reverse order.

Exercise: The 16-bit word 1234H is to be written to address 1FFH. What value will be stored at memory location 1FFH? At which address will the other byte be stored? Write your answer in the form of a table showing the final memory contents:

Address	Data

Which byte enable(s) will need to be asserted to store these values? How many bus cycles will be required? Write out your answers in the form of a table showing the values of the address bus in binary, the values on the data bus in hex, and the values of BHE* and BLE* (H or L) for each bus cycle.

Address	Data Bus	BHE*	BLE*
0001 1111 111x			
0010 0000 000x			

What if the value 12345678H was to be stored at the same address? What if the 16- and 32-bit values were written to address 100H?

Memory and I/O Address Spaces

The Motorola 68000 processors use conventional memory read and write (MOVE) operations to do input and output. Peripheral interfaces appear to the processor as if they were memory locations.

The 80x86 processors can use memory-mapped I/O but can also use special instructions (IN and OUT) for I/O operations. A bus signal (M/IO*) indicates whether a bus cycle is due to a memory or an I/O instruction. These special I/O instructions allow more flexibility in the design of interfaces (e.g. extended cycles for I/O operations). I/O operations can only be done on the first 64kB of the I/O address space.

Bus Control

In order to accommodate slow memory and I/O devices the intel 80x86 processor buses use a READY* input. If the READY* input is not asserted at the end of a T2 processor cycle the 80386SX will generate additional T2 cycle(s) (see below).

Exercise: What signal does the Motorola 68000 use to extend processor cycles?

The W/R* (write/read), D/C* (data/control), and M/IO* (memory/I/O) signals indicate the type of bus cycle being executed (read, interrupt acknowledge or write). The table below shows the possible bus cycles:

[1] From Intel i386SX data sheet.

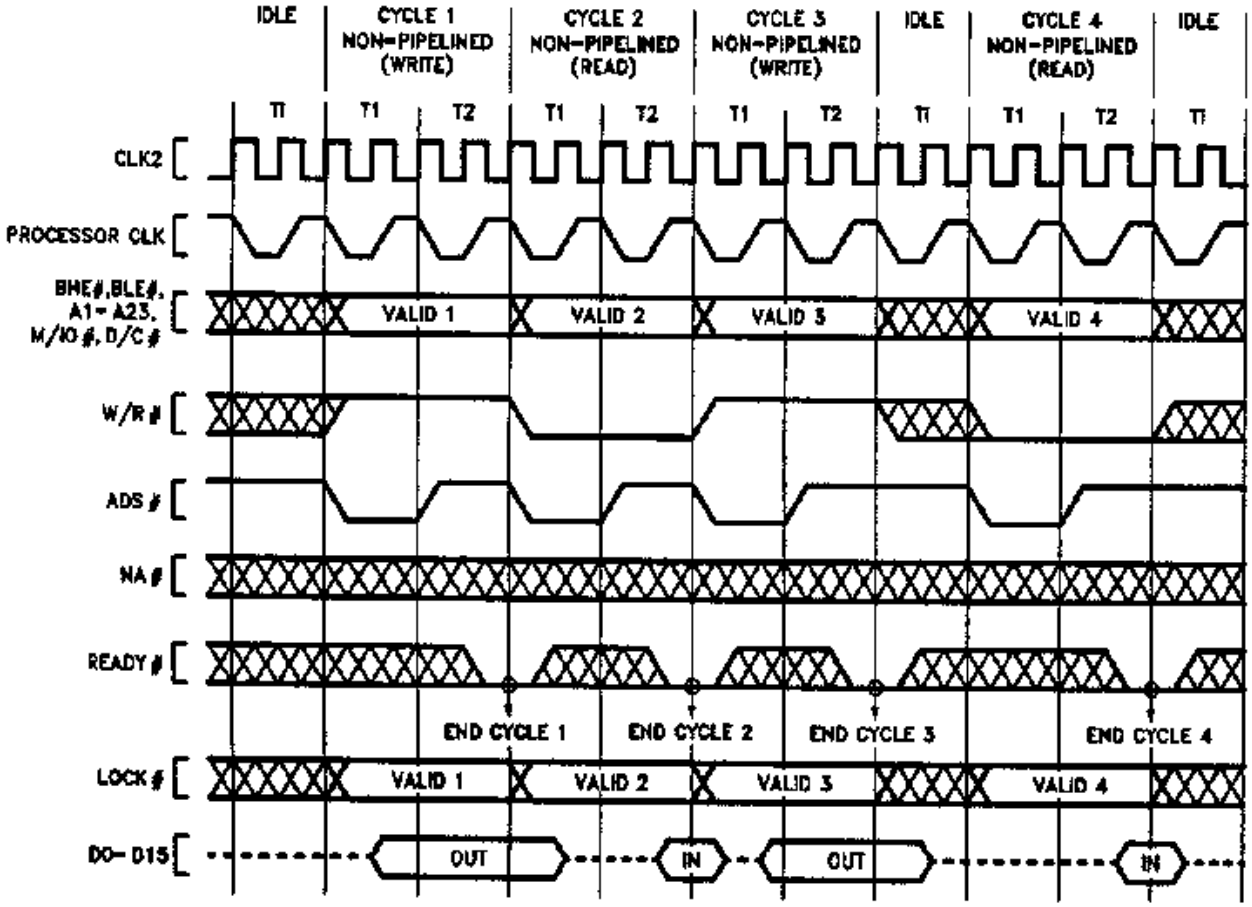


Figure 1: Examples of 80386SX Bus Cycles

D/C*	M/IO*	W/R*	Bus Cycle
H	H	L	memory read
H	H	H	memory write
H	L	L	I/O read
H	L	H	I/O write
L	H	L	instruction fetch
L	L	L	interrupt acknowledge
L	H	H	halt

Another processor bus signal, ADS*, indicates that the contents of the address bus and the three signals above are valid.

Exercise: What are the equivalent signals on the Motorola 68000 processor bus?

Reset and Interrupts

As you might suspect, the RESET input resets the processor. The CPU register contents are reset and the program counter is set so that the CPU will fetch the next instruction from memory location FFFFF0. The memory at this location must therefore contain instructions to restart the system.

The NMI and INTR inputs are used to generate non-maskable and maskable interrupts respectively.

Asserting the NMI input causes the processor to execute the interrupt handler pointed to by an interrupt vector stored in memory.

If interrupts are enabled then asserting INTR causes the CPU to carry out an interrupt acknowledge bus cycle which reads a 1-byte interrupt number from the bus (typically from an interrupt control chip). The corresponding interrupt vector is then fetched and the corresponding interrupt handler executed as with NMI.

In either case the current instruction is completed before the interrupt is recognized. We will cover the

details of the processor's interrupt handling in detail in a later lecture.

Other Signals

The '386SX has a number of other signals which we will not cover at this time. For completeness, these are: HOLD and HOLDA (used by other devices to request that the CPU to give up control of the processor bus by disabling all of its outputs), LOCK* (used to prevent other devices from requesting use of the processor bus), NA* ("next address" used to "pipeline" processor cycles), and PEREQ, BUSY*, and ERROR* (used to interface to a floating point co-processor).

80386SX Bus Cycles

Execution of each 80386SX *instruction* requires one or more bus cycles. Typically, this involves reading an instruction from memory possibly followed by transfers of data between the CPU and memory or I/O devices.

Exercise: What factors will affect the number of bus cycles required to complete an instruction?

In addition to the read and write bus cycles from memory and I/O address space the CPU can also execute an interrupt acknowledge bus cycle and can be in an idle or halted mode.

Read and Write Bus Cycles

The address and bus control signals go active at the start of the T1 processor cycle. During a write cycle the data bus is driven with the value to be written during the second half of T1. During a read cycle the processor loads the value from the data bus at the end of the last T2 cycle.

Input and Output Cycles

I/O read/write cycles are the same as memory read/write cycles except that the M/IO* signal is low.

Interrupt Acknowledge Cycle

An interrupt acknowledge cycle (performed in response to INTR) is the same as a read cycle except that the bus control signals are set to indicate an interrupt acknowledge cycle. The value read during the interrupt acknowledge cycle is then multiplied by 4 and used to load an interrupt vector from this address in memory.

Wait States

At the end of each T2 cycle the processor checks the READY* input. If it is active, the bus cycle is terminated, otherwise and additional T2 cycle is run. These additional *wait states* are used to accommodate slow memory by increasing the time available between when the address is output and the time when the data is required. If the memory being designed into a system will require wait states, a wait state generator circuit must be designed so that READY* is asserted after two or more T2 states have elapsed following the start of the bus cycle.