ASM Functions

While IDE or AHCI Mode is usually used to communicate with storage devices, we implemented ATA PIO Mode for now as the COS kernel doesn’t run in multitasking mode.

The issue with this communication mode is that it fully runs using a CPU bus.
As such, it will freeze execution of other processes until a data transfer completes.

The addressing mode chosen is 48bit LBA mode.
LBA is shorthand for Logical Block Addressing.

In this mode, an ATA compatible drive storage space is divided into Logical Blocks.
A Logical Block is called a **sector** and one sector is exactly 512 bytes long.
Each sector is indexed from 0x0 to 0xFFFFFFFFFFFF (48bit MAX value).

Under PIO Mode, a starting sector is selected. Then, the number of consecutive sectors that may be written to or read from starting from the selected sector must be provided. This number is called the Sector Count.

The maximum number of consecutive sectors that may be written to or read from in a single operation is 0xFFFF (16bit MAX value) sectors. A Sector Count of 0 also means the 16bit MAX value.

Master Disk Detection

In order for the CPU to communicate with an ATA drive, the must first detect a drive connected to an ATA port.
Communication is ensured via the CPU ports 0x1F0 to 0x1F7.
While the ATA specification states that one must select either the Master or the Slave drive (as one ATA port allows two drives to be connected to the computer), Master and Slave drives work in exactly the same way.

The detect_drive label located under ./src/syscalls/read.asm (temporary location) will initialise the CPU bus to communicate with the Master drive.

It first calls the select_master_drive label which does the following: 1) The value 0xA0 is written in the Drive Register (port 0x1F6), selecting the Master Drive (would be 0xB0 for the Slave Drive).
2) The value 0 is written in the LBAlo, LBAmid and LBAhigh Registers (port 0x1F3, 0x1F4, 0x1F5) to initialise them. 3) The value 0 is written in the Sector Count Register (port 0x1F2) to initialise it. 4) The command 0xEC (IDENTIFY Command) is written in the Command Register (port 0x1F7 in write mode). 5) Finally, the Status Register is read (port 0x1F7 in read mode).
If value 0 is returned, the Master drive does not exist.

Otherwise, the label wait_for_ready is then called.
The Status Register’s value is an 8bit value representing the drive’s current status and each bit correspond to a flag.
The function polls Status Register until Bit 7 (BSY bit) is cleared as while it is set, the drive will not process any command.

The check_non_ata_status_ports label will check values currently stored in the LBAmid and LBAhigh Registers (ports 0x1F4 and 0x1F5). If at least one of them isn’t set to 0, it means that the drive doesn’t support ATA PIO Mode.

The CPU will execute instructions written under the label wait_for_information. The function polls the Status Register (port 0x1F7) until either of these bits are set: - Bit 0 is the ERR bit (ERRor). If set, an error occured during a command execution. - Bit 3 is the DREQ bit (Data REQuest). If set, the drive wants to exchange data with the host controller.
The function reads the Status Register until Bit 3 is set, after which it returns.

Whenever data is sent from the drive to the CPU, the Data Register (port 0x1F0) is filled with 256 words.
A word is a 16bit value (2 bytes long).
The read_loop label has the ability to read these words and is called as the final step of this drive detection procedure.

The loop instruction will jump to the read_loop.loop label each time it is called while decrementing the value stored in the rcx register.
As the detect_drive label is called from C code, the rdi register is its first argument according to the C standard.
The rdi register stores an address to a table containing 256 words.

Once all words have been read from the Data Register (port 0x1F0), the function wait_for_ready is called again until the drive isn't busy anymore.

The retrieved words sent by the drive after the IDENTIFY command is used returns information on the drive itself, mosr notably: - Bit 10 of word 83 allow for the detection of 48bit LBA support. - A 64bit value stored as 4 words (word 100 to 103) represents the maximum number of sectors supported by the selected drive.
Using this value, the kernel may therefore know the storage capacity of the ATA Drive during its initialisation.

Read Function

The read_byte label is used as the low level read function using 48bit LBA mode.
When called from C code, the first argument is a table of 256 words (stored in the rdi register), the second argument is an LBA value passed as a 64bit value (stored in the rsi register) and the third argument is the 16bit sector count (stored in the rdx register).

A total of 8 bytes have to be provided to initialise the read command as the Sector Count is a word (2 bytes) and the LBA mode is 48bits (6 bytes).

The high byte of the Sector Count must be sent to the Sector Count Register (port 0x1F2) followed by the three high bytes of the LBA (byte 6, byte 5 and byte 4) to the LBAhigh, LBAmid and LBAlo Registers. Then, the low bytes of the Sector Count and LBA are sent.

The read_byte function executes the following procedure: 1) As the out instruction must be used with the dx register, the value stored in rdx as the third argument is first copied into the rcx register. 2) The value 0x40 is written into the Drive Register (port 0x1F6) to select the Master Drive (value 0x50 for the Slave Drive). 3) The Sector Count high byte is then extracted from the rcx register via the ch register and then written into the Sector Count Register (port 0x1F2). 4) Byte 4 of the LBA is written in the LBAlo Register (port 0x1F3). 5) Byte 5 of the LBA is written in the LBAmid Register (port 0x1F4). 6) Byte 6 of the LBA is written in the LBAhigh Register (port 0x1F5). 7) The Sector Count low byte is then extracted from the rcx register via the cl register and then written into the Sector Count Register (port 0x1F2). 8) Byte 1 of the LBA is written in the LBAlo Register (port 0x1F3). 9) Byte 2 of the LBA is written in the LBAmid Register (port 0x1F4). 10) Byte 3 of the LBA is written in the LBAhigh Register (port 0x1F5). 11) Then, the 0x24 (READ SECTORS EXT) command is written into the Command Register (port 0x1F7 in write mode). 12) The function then calls the wait_for_ready label which polls the Status Register (port 0x1F7 in read mode) until the BSY bit is cleared. 13) Finally, the read_loop function described in the Master Disk Detection section is called, reading 256 words at the selected sector at the address stored in the rdi register.

As the low level read function isn't yet associated to the Ext4 filesystem module, it may only be called from C code with a sectorcount of 1.

Write Function

Similarly to the read_byte function, the write_byte function is used as the low level write function using 48bit LBA mode.

When called from C code, the first argument is a table of 256 words (stored in the rdi register), the second argument is an LBA value passed as a 64bit value (stored in the rsi register) and the third argument is the 16bit sector count (stored in the rdx register).

A total of 8 bytes have to be provided to initialise the read command as the Sector Count is a word (2 bytes) and the LBA mode is 48bits (6 bytes).

The high byte of the Sector Count must be sent to the Sector Count Register (port 0x1F2) followed by the three high bytes of the LBA (byte 6, byte 5 and byte 4) to the LBAhigh, LBAmid and LBAlo Registers. Then, the low bytes of the Sector Count and LBA are sent.

The write_byte function executes the following procedure: 1) As the out instruction must be used with the dx register, the value stored in rdx as the third argument is first copied into the rcx register. 2) The value 0x40 is written into the Drive Register (port 0x1F6) to select the Master Drive (value 0x50 for the Slave Drive). 3) The Sector Count high byte is then extracted from the rcx register via the ch register and then written into the Sector Count Register (port 0x1F2). 4) Byte 4 of the LBA is written in the LBAlo Register (port 0x1F3). 5) Byte 5 of the LBA is written in the LBAmid Register (port 0x1F4). 6) Byte 6 of the LBA is written in the LBAhigh Register (port 0x1F5). 7) The Sector Count low byte is then extracted from the rcx register via the cl register and then written into the Sector Count Register (port 0x1F2). 8) Byte 1 of the LBA is written in the LBAlo Register (port 0x1F3). 9) Byte 2 of the LBA is written in the LBAmid Register (port 0x1F4). 10) Byte 3 of the LBA is written in the LBAhigh Register (port 0x1F5). 11) Then, the 0x34

(WRITE SECTORS EXT) command is written into the Command Register (port 0x1F7 in write mode). 12) The function then calls the wait_for_ready label which polls the Status Register (port 0x1F7 in read mode) until the BSY bit is cleared. 13) The write_loop function is called, writing 256 words at the selected sector at the address stored in the rdi register.
The loop instruction will jump to the write_loop.loop label each time it is called while decrementing the value stored in the rcx register.
The rdi register stores an address to a table containing 256 words.
The value written into the ax register is then written into the Data Register register (port 0x1F0).
As a delay is required to write the value to disk, a jmp to the write_loop.delay label is made before the next value is written to the Data Register. 14) Finally, the complete_write_command label is called. Every "WRITE SECTORS EXT" (0x34) command must be ended by a "Cache Flush" command (0xE7).

As the low level write function isn't yet associated to the Ext4 filesystem module, it may only be called from C code with a sectorcount of 1.

External Documentation

- ATA Mode Documentation
- Status Register Bits