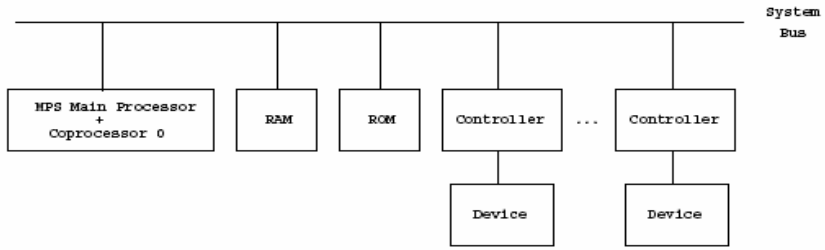
Laboratorio di Sistemi Operativi
Anno Accademico 2006-2007

uMPS Introduction
Part 3
Mauro Morsiani

uMPS processor architecture

- **The uMPS architecture**



uMPS devices and I/O management

- **uMPS devices and interfaces:**
  - uMPS provides a basic set of devices:
    - TOD clock and interval timer
    - disks
    - tapes
    - ethernet-like network interfaces
    - printers
    - tty-like terminals
  - each *device class* is associated to a specific *interrupt line*
  - for each device class (not **TOD**) up to 8 devices may be installed
  - each device is operated by a *controller*
  - the processor talks with controllers using *device registers*
  - device registers are locations in memory
  - *all addresses in I/O management are physical memory addresses*

uMPS devices and I/O management

- **Device register structure:**
  - Each device register has a *base address*
  - All device registers (except **TOD**) have the same basic structure
  - Device register commands and status codes for each device class are (obviously) different, but share a common operation logic

Field #	Address	Field Name
0	(base) + 0x0	**STATUS**
1	(base) + 0x4	**COMMAND**
2	(base) + 0x8	**DATA0**
3	(base) + 0xc	**DATA1**

uMPS devices and I/O management

- **Device registers structure (cont'd):**
  - only some device registers are writable
  - device registers are writable only when device is idle
  - device registers "freeze" when an operation is in progress
- **uMPS processor-device communication protocol:**
  - is a *full-handshake interrupt-driven* protocol
  - *full-handshake*: each exchange has to be acknowledged explicitly
  - *interrupt-driven*: interrupts are used to notify the CPU that something has happened

uMPS devices and I/O management

- **uMPS processor-device communication protocol details:**
  - the CPU initiates an operation by writing a command in a device register
  - the device starts the operation and sets some "working…" status code in the device register
  - the device completes the operation and:
    - sets some "completed…" status code in the device register
    - raises an interrupt by *asserting* (setting on) an interrupt line
  - the CPU:
    - may check the operation outcome (by looking at the status code in the device register)
    - may acknowledge it (by issuing an acknowledge command): this *deasserts* (turns off) the interrupt line
  - An optimization: a new command implicitly acknowledges a previous operation

uMPS devices and I/O management

- **Interrupt line vs. device class:**
  - higher line # : lower speed/priority

Interrupt Line #	Device Class
2	Bus (Interval Timer)
3	Disk Devices
4	Tape Devices
5	Network (Ethernet) Devices
6	Printer Devices
7	Terminal Devices

uMPS devices and I/O management

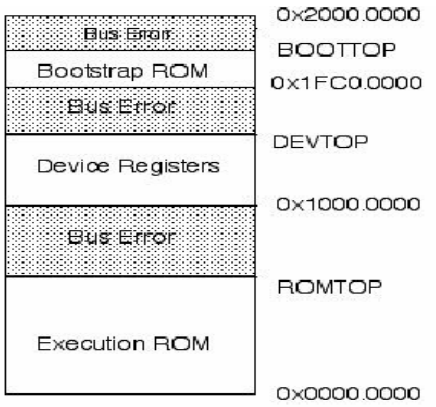
- **How to handle a lot of devices:**
  - **TOD/Interval Timer** + (5 lines x 8 devices) = 41 possible devices
  - only 6 interrupt lines
  - How to know which devices are installed?
  - How to understand which devices have raised an interrupt?
  - Where to put all these device registers?

uMPS devices and I/O management

- **How to handle a lot of devices (cont'd):**
  - How to know which devices are installed?
    - with an *Installed Devices Bit Map* (a 1 in the map means the device is present)
  - How to understand which devices have raised an interrupt?
    - with an *Interrupting Devices Bit Map* (a 1 in the map means the device is raising an interrupt)
    - This also means: an interrupt line will remain asserted until *all* device interrupts pending on that line get acknowledged

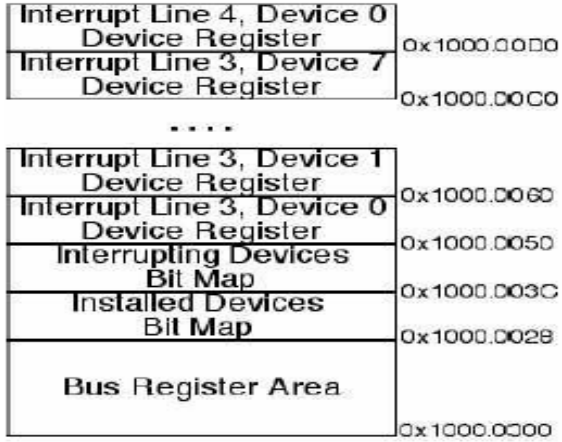
uMPS memory management

- **Where to put all these device registers?**



uMPS devices and I/O management

- **Device Register Area details:**



uMPS devices and I/O management

- **Device Register Area details (cont'd):**
  - **Bus Register Area: TOD/Interval Timer** and and other useful information (**RAMSIZE**, etc.)

Physical Address	Field Name
0x1000.0000	RAM Base Physical Address
0x1000.0004	Installed RAM Size
0x1000.0008	Exec. ROM Base Physical Address
0x1000.000c	Installed Exec. ROM Size
0x1000.0010	Bootstrap ROM Base Physical Address
0x1000.0014	Installed Bootstrap ROM Size
0x1000.0018	Time of Day Clock - High
0x1000.001c	Time of Day Clock - Low
0x1000.0020	Interval Timer
0x1000.0024	Time Scale

uMPS devices and I/O management

- **Device Register Area details (cont'd):**
  - Special device registers in the **Bus Register Area**:
    - **TOD** (**TODHI** + **TODLO**) (64 bit, readonly): number of clock ticks from last boot/reset
    - **Interval Timer** (read/write):
      - starts from 0xFFFF.FFFF at boot/reset
      - gets decremented by 1 on each clock tick
      - raises an interrupt on line 2 when an underflow happens (0x0000.0000 → 0xFFFF.FFFF transition)
    - **Time Scale** (readonly): number of clock ticks per microsecond

uMPS devices and I/O management

- **Installed Devices Bit Map and format:**
  - Devices are numbered 0..7

Word #	Physical Address	Field Name
0	0x1000.0028	Interrupt Line 3 Installed Devices Bit Map
1	0x1000.002C	Interrupt Line 4 Installed Devices Bit Map
2	0x1000.0030	Interrupt Line 5 Installed Devices Bit Map
3	0x1000.0034	Interrupt Line 6 Installed Devices Bit Map
4	0x1000.0038	Interrupt Line 7 Installed Devices Bit Map

uMPS devices and I/O management

- **Interrupting Devices Bit Map and format:**
  - Devices are numbered 0..7

Word #	Physical Address	Field Name
0	0x1000.003C	Interrupt Line 3 Interrupting Devices Bit Map
1	0x1000.0040	Interrupt Line 4 Interrupting Devices Bit Map
2	0x1000.0044	Interrupt Line 5 Interrupting Devices Bit Map
3	0x1000.0048	Interrupt Line 6 Interrupting Devices Bit Map
4	0x1000.004C	Interrupt Line 7 Interrupting Devices Bit Map

uMPS devices and I/O management

- **Where is my device?**
  - Given a device identified by:
    - an interrupt line (*IntLineNo*)
    - a device number (*DevNo*)
  - The base address of the corresponding device register is:

devAddrBase = 0x1000.0050 + ((IntlineNo - 3) * 0x80) + (DevNo * 0x10)

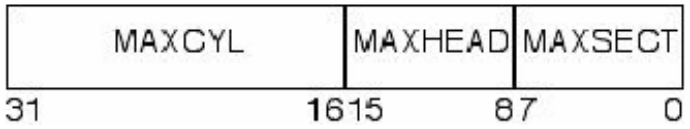
uMPS devices and I/O management

- **Device register structure:**

Field #	Address	Field Name
0	(base) + 0x0	**STATUS**
1	(base) + 0x4	**COMMAND**
2	(base) + 0x8	**DATA0**
3	(base) + 0xc	**DATA1**

uMPS devices and I/O management

- **Disk device:**
  - DMA (*Direct Memory Access*) capable
  - 4 KB block size
  - up to 64K cylinders, 256 heads, 256 sectors
  - *disk geometry*: Cylinder/Head/Sector (CHS)
  - geometry packed in **DATA1** field
  - numbered starting from 0, up to MAX{CYL/HEAD/SECT} -1



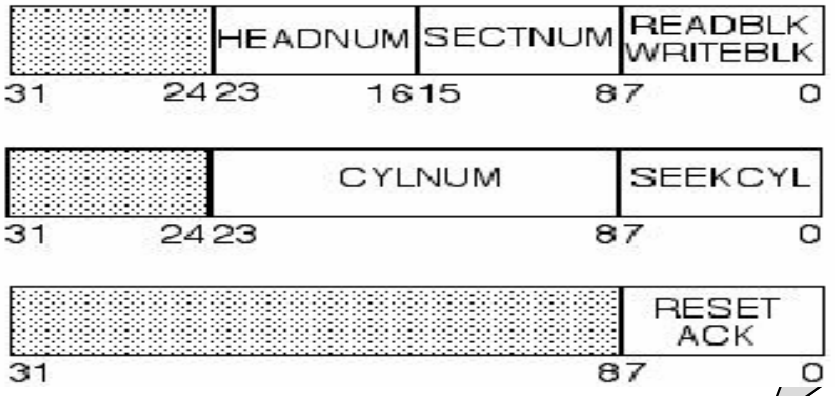
uMPS devices and I/O management

- **Disk device commands:**

Code	Command	Operation
0	RESET	Reset the device and move the boom to cylinder 0
1	ACK	Acknowledge a pending interrupt
2	SEEKCYL	Seek to the specified **CYLNUM**
3	READBLK	Read the block located at (**HEADNUM**, **SECTNUM**) in the current cylinder and copy it into RAM starting at the address in **DATA0**
4	WRITEBLK	Copy the 4KB of RAM starting at the address in **DATA0** into the block located at (**HEADNUM**, **SECTNUM**) in the current cylinder

uMPS devices and I/O management

- **Disk device COMMAND field format:**



uMPS devices and I/O management

- **Disk device STATUS codes:**

Code	Status	Possible Reason for Code
0	Device Not Installed	Device not installed
1	Device Ready	Device waiting for a command
2	Illegal Operation Code Error	Device presented unknown command
3	Device Busy	Device executing a command
4	Seek Error	Illegal parameter/hardware failure
5	Read Error	Illegal parameter/hardware failure
6	Write Error	Illegal parameter/hardware failure
7	DMA Transfer Error	Illegal physical address/hardware failure

uMPS devices and I/O management

- **Tape device:**
  - DMA capable
  - *read-only*
  - employed to load programs and data files
  - 4 KB block size
  - **DATA1**: tape marker under head
  - no tape: **DATA1** = end-of-tape (EOT)

uMPS devices and I/O management

- **Tape device DATA1 codes:**
  - Tape = *TS* EOB EOB .. *EOF* EOB EOB ..*EOF* EOB EOB.. *EOT*
  - EOF stands also for EOB
  - EOT stands also for EOF

Code	Marker	Meaning
0	**EOT**	End of Tape
1	**EOF**	End of File
2	**EOB**	End of Block
3	**TS**	Tape Start

uMPS devices and I/O management

- **Tape device commands:**

Command	Code	Operation
RESET	0	Reset the device and rewinds the tape to start
ACK	1	Acknowledge an interrupt request
SKIPBLK	2	Skip current block, reaching the following marker
READBLK	3	Read current block, reaching the following marker; copy it to RAM starting at **DATA0** physical address
BACKBLK	4	Go back one block, reaching the previous marker

uMPS devices and I/O management

- **Tape device STATUS codes:**

Code	Status
0	Device Not Installed
1	Device Ready
2	Illegal Operation Code Error
3	Device Busy
4	Skip Error
5	Read Error
6	Back 1 Block Error
7	DMA Transfer Error

uMPS devices and I/O management

- **Network device:**
  - DMA capable
  - 1514-byte Ethernet frame size and buffer format
  - 6-byte MAC (*Media Access Control*) address
  - **DATA0** used for buffer address
  - **DATA1** reports or sets packet length
  - non-blocking reads (**DATA1** == 0)
  - several *operation modes*

uMPS devices and I/O management

- **Network device operation modes:**
  - *normal / promiscuous (0x4) mode*:
    - **normal**: only broadcast and packets explicitly directed to the interface are listened to, others are ignored
    - **promiscuous**: all packets are listened to
  - *polling / interrupt driven (0x2) mode*:
    - **polling**: packet reads must be explicit (no interrupts)
    - **interrupt driven**: receiving a packet raises an interrupt
  - *named (0x1) / unnamed mode*:
    - **named**: transmitted frames are tagged with the MAC address of the device
    - **unnamed**: the device does not modify the MAC address provided for transmitted frames

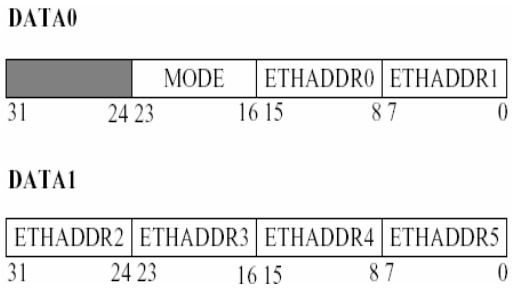
uMPS devices and I/O management

- **Network device commands:**

Command	Code	Operation
RESET	0	Reset the device
ACK	1	Acknowledge an interrupt request
READCONF	2	Read the current configuration
CONFIGURE	3	Configure the interface
READNET	4	Read a Packet from the Network Interface
WRITENET	5	Write a Packet on the Network Interface

uMPS devices and I/O management

- **Network device DATA0 and DATA1 layout in CONFIGURE and READCONF:**



uMPS devices and I/O management

- **Network device STATUS codes:**

Code	Status
0	Device Not Installed
1	Device Ready
2	Illegal Operation Code Error
3	Device Busy
4	(unused for network)
5	Read Error
6	Write Error
7	DMA Transfer Error

uMPS devices and I/O management

- **Printer device:**
  - simple parallel line printer
  - character-based
  - **DATA0**: character to be printed
  - **DATA1**: unused

uMPS devices and I/O management

- **Printer device commands:**

Code	Command	Operation
0	RESET	Reset the device interface
1	ACK	Acknowledge a pending interrupt
2	PRINTCHR	Transmit the character in **DATA0** over the line

uMPS devices and I/O management

- **Printer device STATUS codes:**

Code	Status	Possible Reason for Code
0	Device Not Installed	Device not installed
1	Device Ready	Device waiting for a command
2	Illegal Operation Code Error	Device presented unknown command
3	Device Busy	Device executing a command
4	Print Error	Error during character transmission

uMPS devices and I/O management

- **Terminal device:**
  - serial line terminal for basic I/O interactions
  - 8-bit character
  - two sub-devices: *transmitter* and *receiver*
  - *both* sub-devices will raise interrupts

Field #	Address	Field Name
0	(base) + 0x0	**RECV_STATUS**
1	(base) + 0x4	**RECV_COMMAND**
2	(base) + 0x8	**TRANSM_STATUS**
3	(base) + 0xc	**TRANSM_COMMAND**

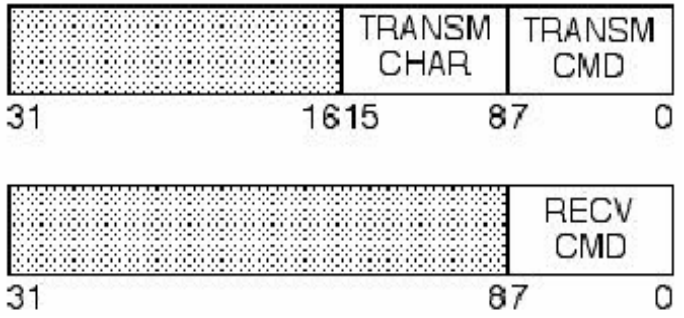
uMPS devices and I/O management

- **Terminal device commands:**

Code	TRANSM COMMAND	RECV COMMAND	Operation
0	RESET	RESET	Reset the transmitter or receiver interface
1	ACK	ACK	Ack a pending interrupt
2	TRANSMITCHAR	RECEIVECHAR	Transmit or Receive the character over the line

uMPS devices and I/O management

- **Terminal sub-device COMMAND format:**



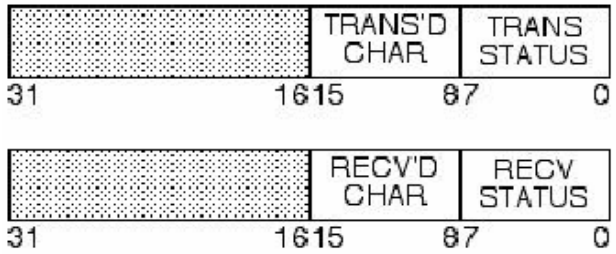
uMPS devices and I/O management

- **Terminal sub-device STATUS codes:**

Code	RECV STATUS Meaning	TRANSM STATUS Meaning
0	Device Not Installed	Device Not Installed
1	Device Ready	Device Ready
2	Illegal Operation Code Error	Illegal Operation Code Error
3	Device Busy	Device Busy
4	Receive Error	Transmit Error
5	Character Received	Character Transmitted

uMPS devices and I/O management

- **Terminal sub-device STATUS format:**



uMPS devices and I/O management

- **Things to remember:**
  - First write parameters (**DATA0**, **DATA1**) *then* **COMMAND**
  - Check **STATUS** after operations
  - Remember always to ACK commands
  - Beware of DMA