Devices and Device Interfacing

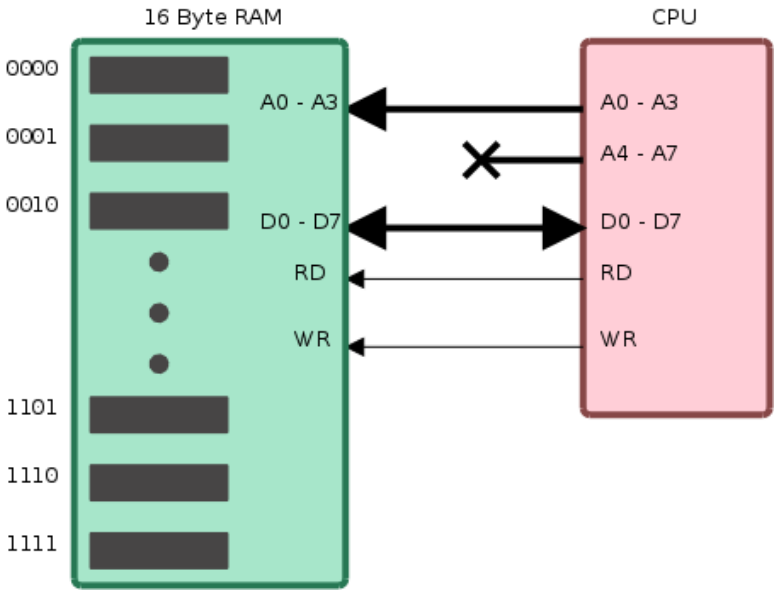
Zilogic Systems

1. Memory Interfacing

1.1. Bus Basics

- Bus - a group of signals used as a shared communication medium among devices.
- Bus that connects CPU to Memory - System Bus or Memory Bus.
- Bus has 3 different sets of signals
  - Address
  - Data
  - Control
- Interfacing single memory chip.
- RD/WR are control signals.

Figure 1. Interfacing single memory chip



1.2. Bus Transactions

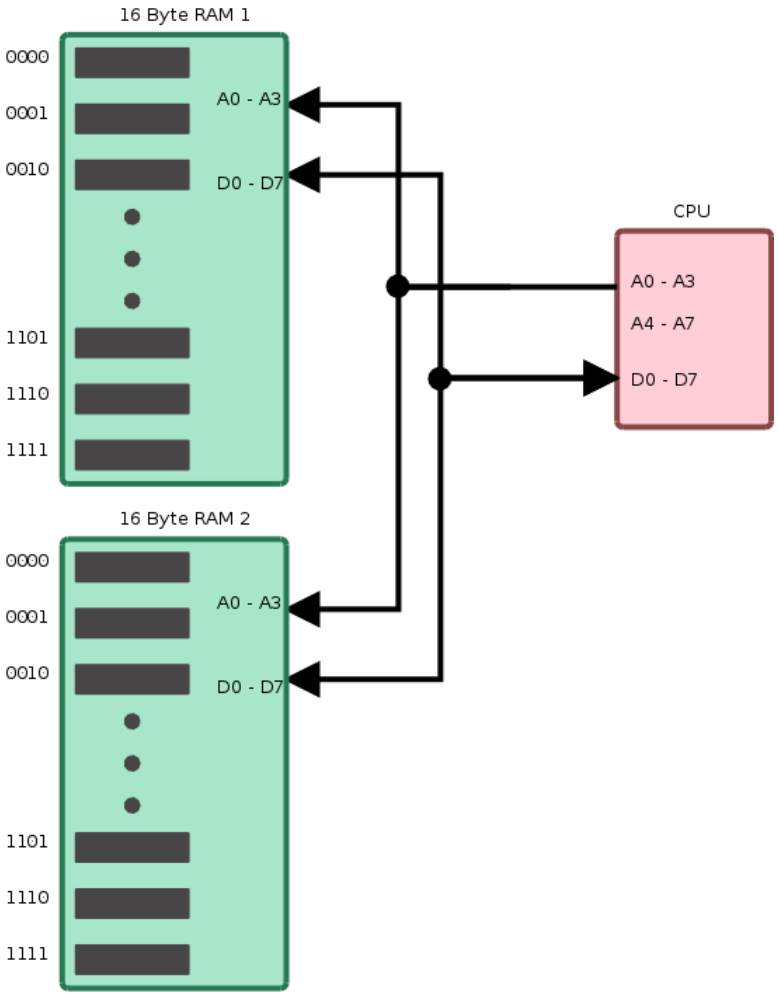
- Sequence of actions required to perform a read/write is called a transcation.
- Bus transactions involve two devices - Master and Slave.
- Master initiates the transaction, and slave responds to the master.
- Read Transaction
  1. The bus master drives the address.
  2. The RD is pulsed.
  3. The slave drives the data.
- Write Transaction
  1. The bus master drives the address and data.

2. The WR is pulsed.
3. The slave samples the data from the bus using the WR pulse.

1.3. Address Decoding

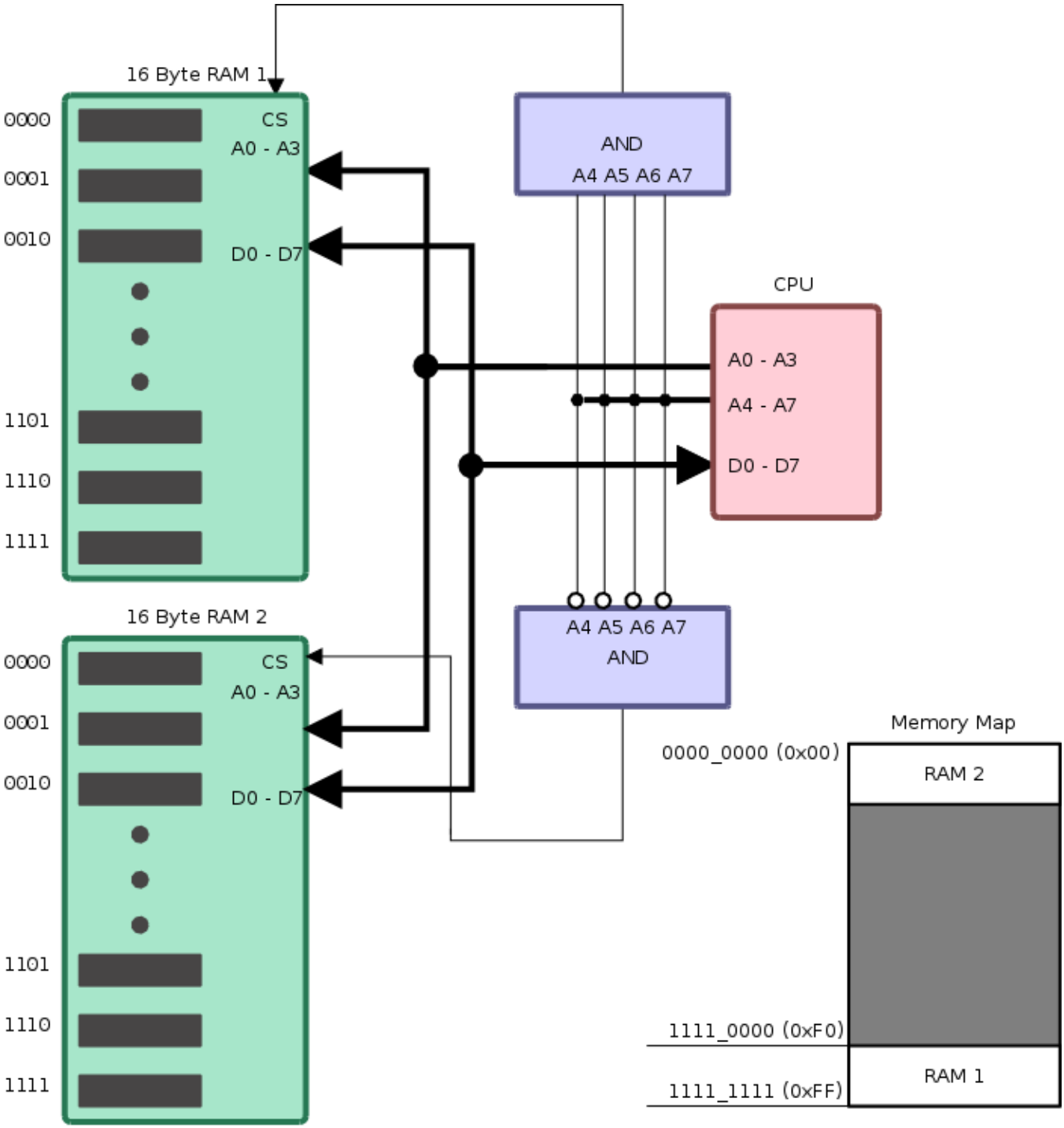
- Interfacing two memory chips.
- When the address and data is driven by the CPU, which chip is supposed to receive it?

Figure 2. Interfacing two memory chips



- Lower address lines decide which memory location within the chip.
- Higher address line are used to decide which chip.
- Devices usually have a chip select signal.
- Only when the chip select is enabled the chip will respond.
- Higher address lines are passed through a combinational logic to generate a chip select.
- Combinational logic is called the decoding logic.
- The decoding logic should ensure that the device addresses do not overlap with any other device.

Figure 3. Interfacing two memory chips with decoding



2. Device Communication

1. Register Access
2. Interrupts

3. Register Access

- Example device - 16 digit, 7 Segment display.
- Has 16 registers. Each register controls a digit.

Figure 4. 16 Digit, 7 segment display

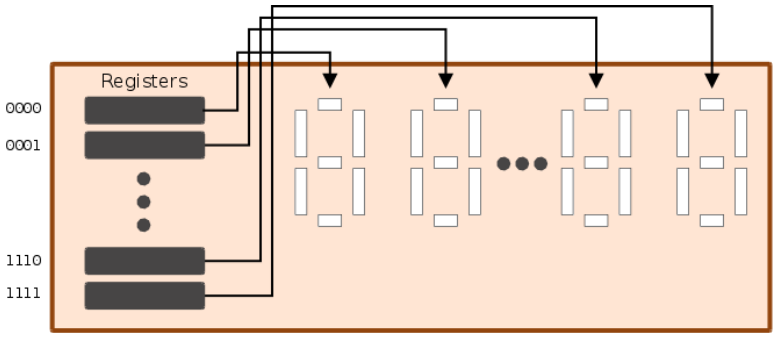
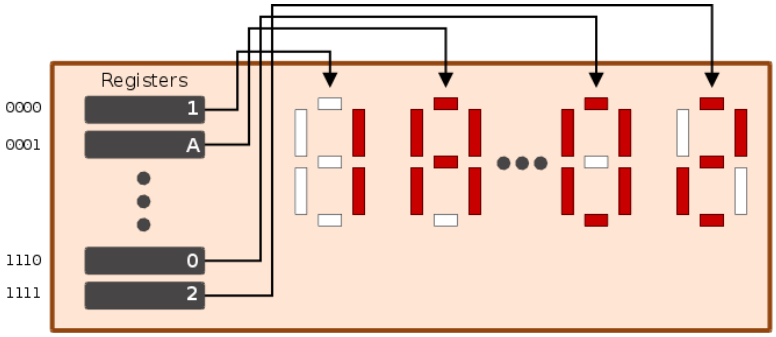


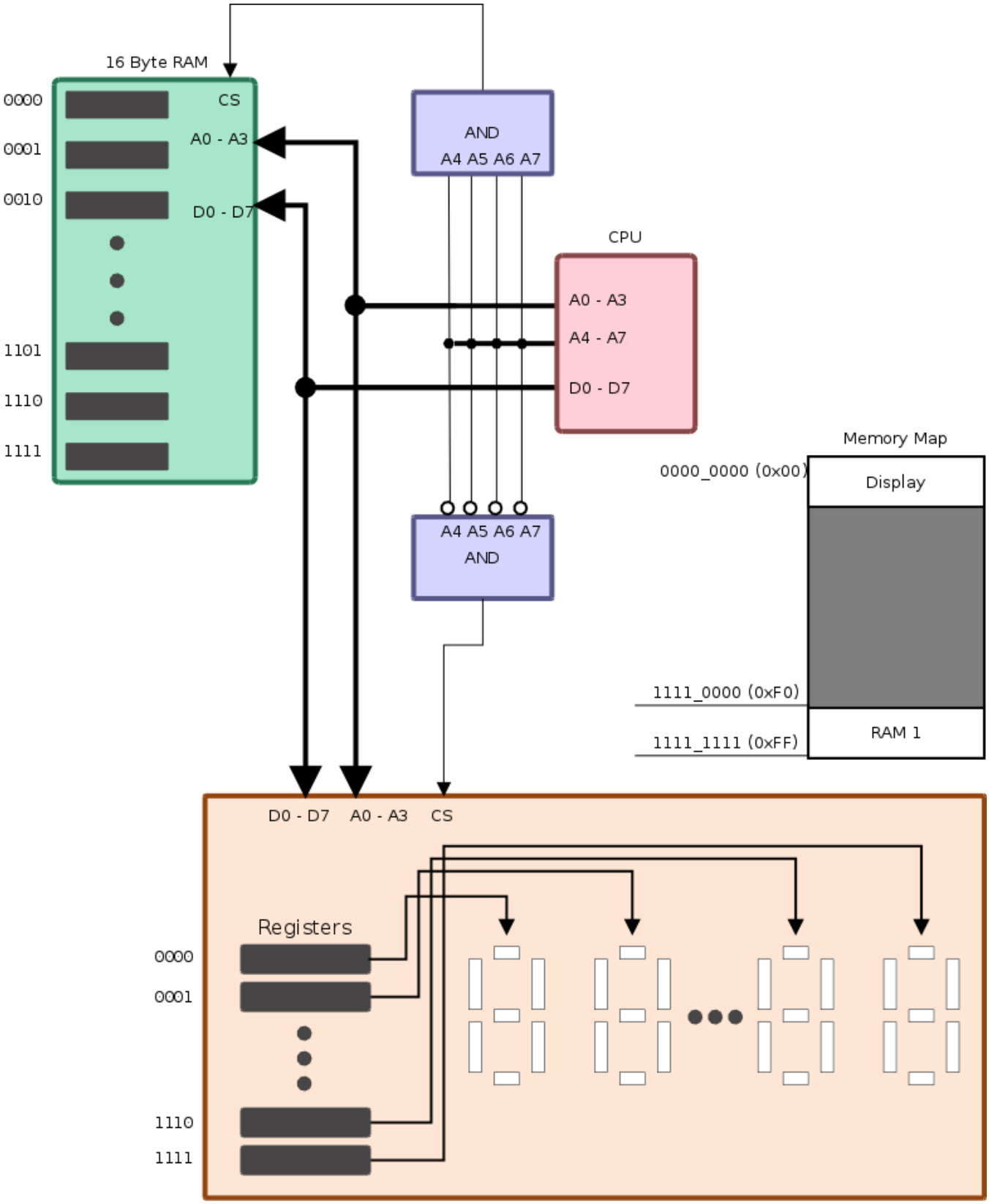
Figure 5. 16 Digit, 7 segment display displaying data



4. Memory Mapped I/O

- Devices can be directly connected to the system bus.
- When a read or write transaction occurs, the device registers are read and written to.
- Interfacing memory and display.
- Problems with Memory Mapped I/O
  - Compiler Optimization
  - System Level Optimization - Cache
  - Processor Optimization - access re-ordering, pre-fetching
- Disable optimization
- Alternate method: Separate space for I/O - Port mapped I/O

Figure 6. Memory Mapped Display

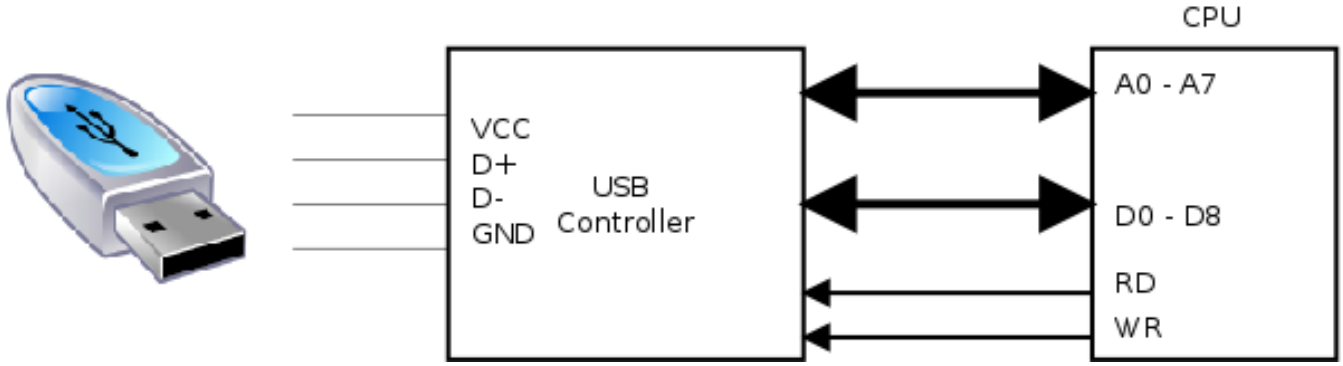


5. I/O with Controllers

- Sometimes the devices are not memory mapped or port mapped.

- The device is connected to a controller which in turn is mapped in the processors address space.
- The CPU writes to the controller's registers and the controller generates appropriate message to the devices.
- Example: USB devices, I²C devices, serial devices.

Figure 7. USB Controller

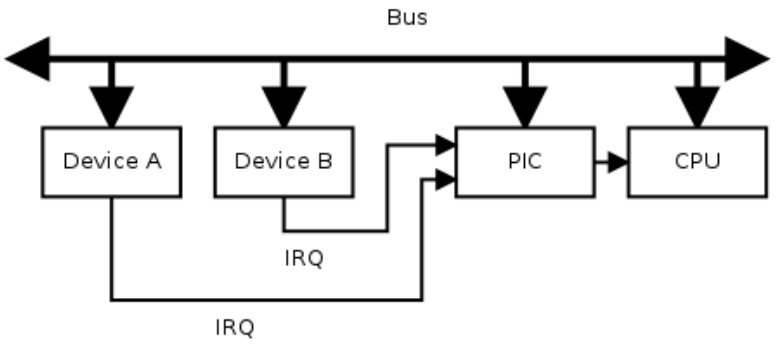


6. Interrupts

- Hardware interrupt is an asynchronous signal from hardware indicating the need for attention. — Wikipedia
- For example, a keyboard device might raise an interrupt when a key is pressed.
- Without an interrupt the processor would have to keep repeatedly checking if a key has been pressed by reading the device's registers, and will not be able to do other activities.
- An interrupt causes the CPU to stop executing the current program and start executing a special piece of code called the interrupt handler.

7. Programmable Interrupt Controller

Figure 8. PIC and IRQs

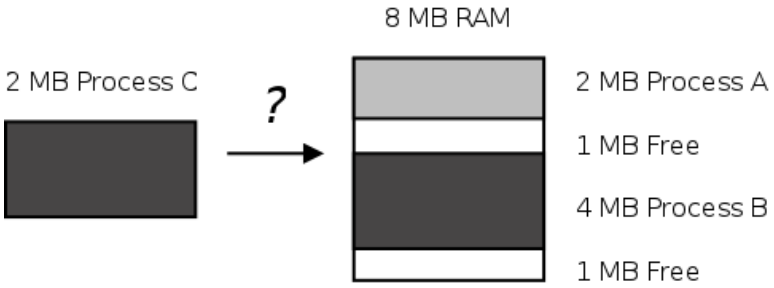


- Processors usually have only 1 interrupt line,
- What if multiple devices want to interrupt the processor?
- Devices send IRQs to a PIC.
- The PIC's goal is to provide prioritized interrupt capabilities to the main processor (CPU) through a single line. When a device issues an interrupt, it is delivered to one of the PIC's IRQs (pins), from there the interrupt is generated to the CPU, which, in turn, checks with the interrupt controller for the source of the interrupt.

8. MMU

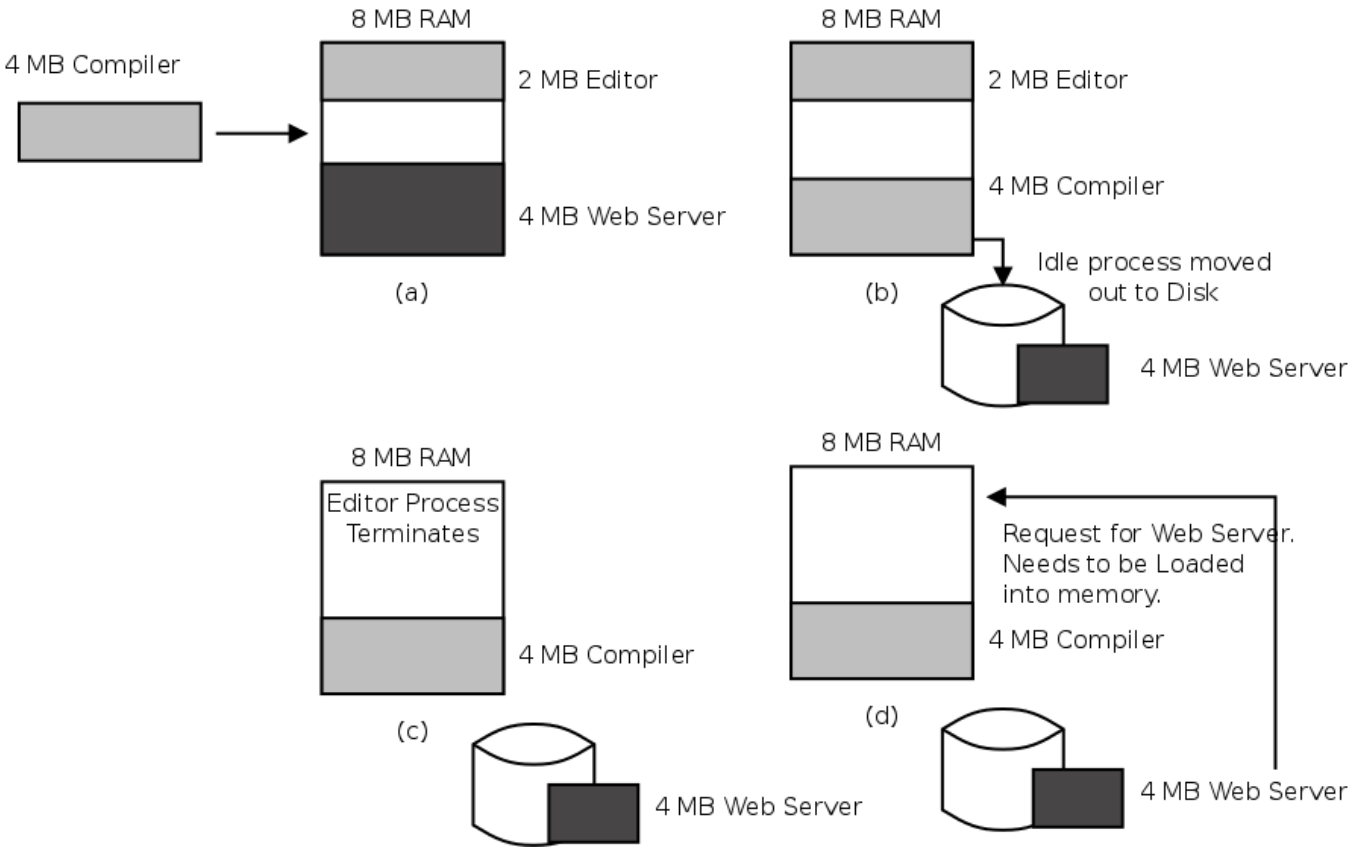
- Multitasking OS - tries to create an virtual computer for each process. So that each process thinks that it is the only task running in the computer.
- One process should not interfere with another process — **protection**.
- As system allocates and deallocates memory, the memory gets fragmented, some mechanism is required to overcome **fragmentation**.

Figure 9. Fragmentation



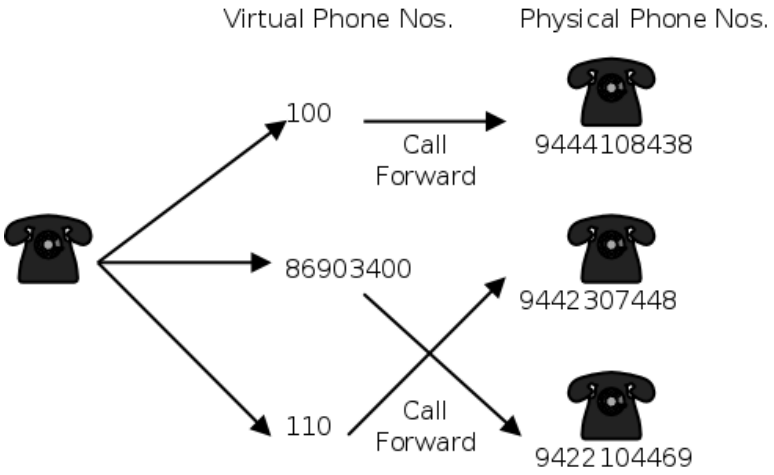
- It is also useful to be able to move out idle process to secondary storage, so that the memory can be used by active processes. This is called **swapping**. Doing this without the knowledge of the process can be tricky.

Figure 10. Swapping



- Call Forwarding - Virtual Phone Nos, Physical Phone Nos.

Figure 11. Call Forwarding



- CPU generates virtual addresses. MMU translates virtual address to physical address. Physical address is used to access memory.
- Translation is not done on byte basis, it is done for chunk of contiguous bytes. This chunk of bytes is called page.
- A table mapping virtual pages to physical pages is called page translation table.
- It is not necessary that all virtual addresses should have a valid mapping to a physical addresses.
- If a mapping is not present for a virtual address, that the CPU tries to access, a **page fault** interrupt is generated.
- To acheive protection, each process is provided with its own page translation table. When a process is executing, the page translation table of the process is loaded into the MMU. With this setup it is impossible for one process to access the memory locations of another process.

Figure 12. Protection with MMU

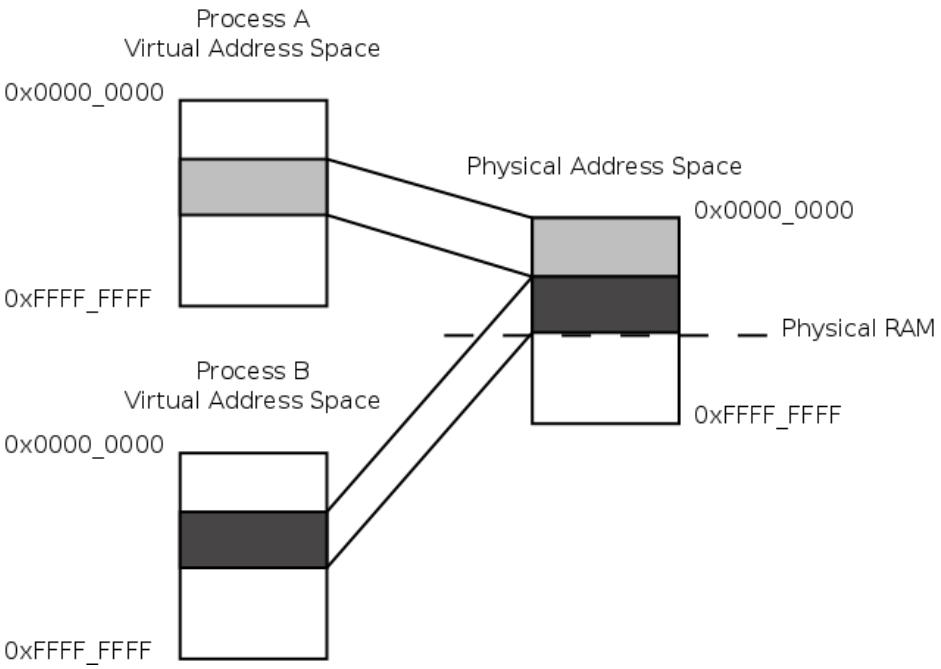


Figure 13. Fragmentation with MMU

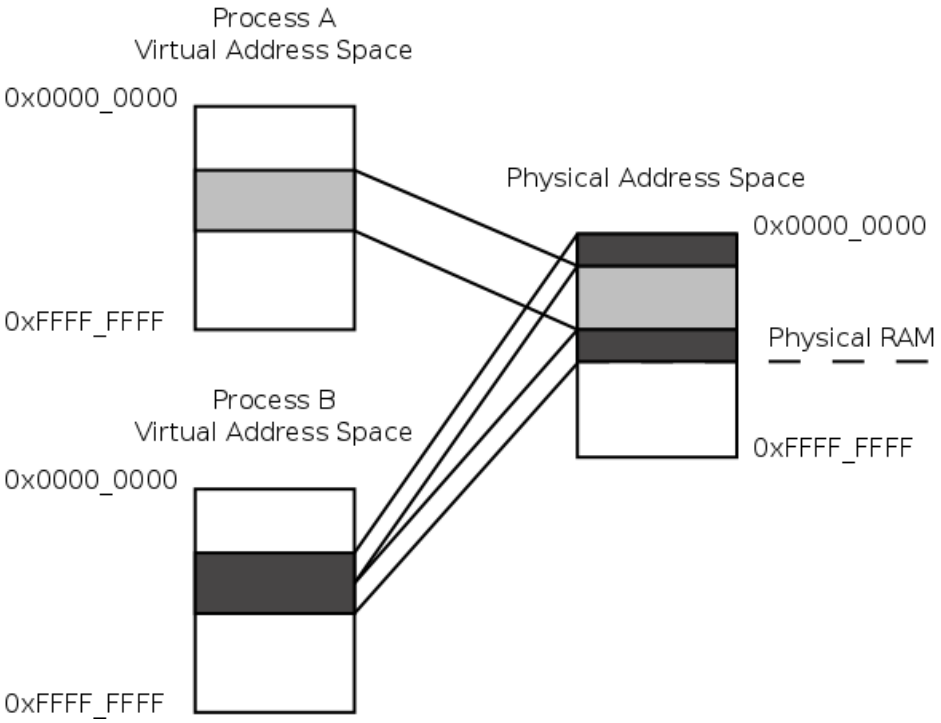
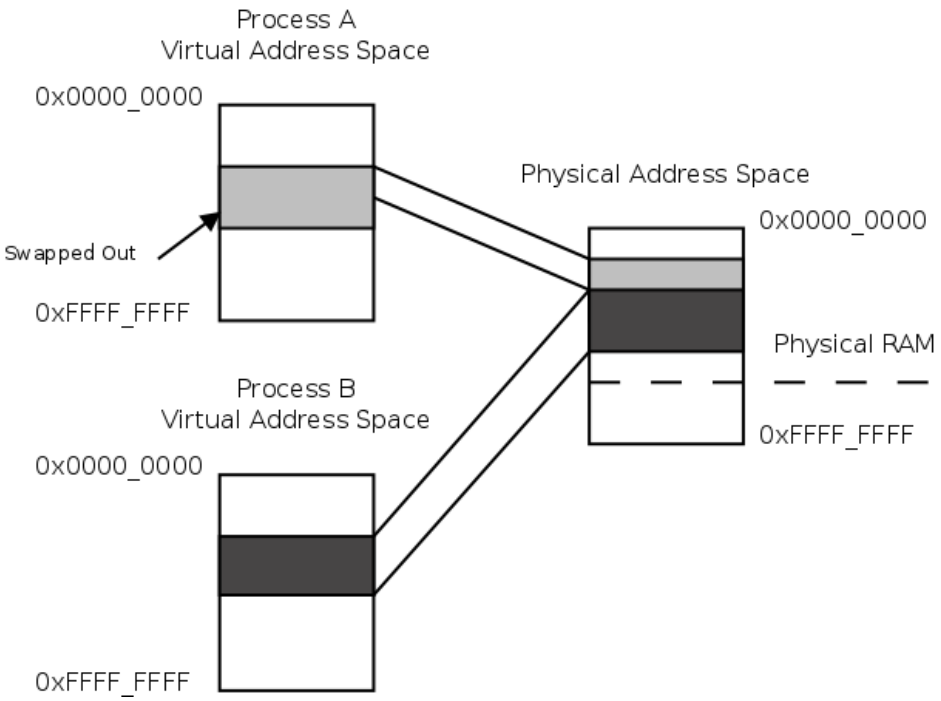
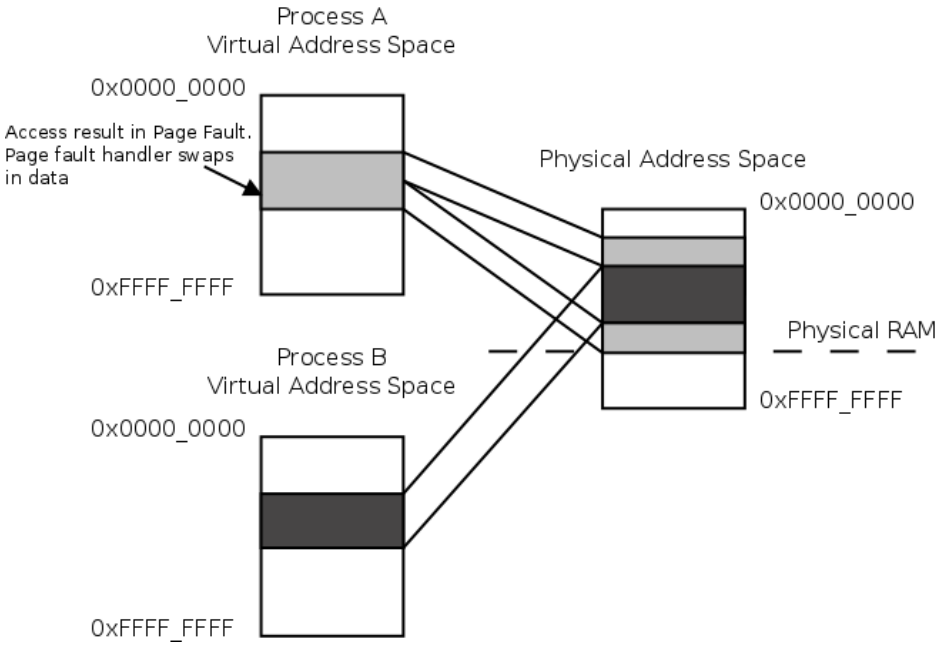


Figure 14. Swaping with MMU




Process A
Virtual Address Space
0x0000_0000
Access result in Page Fault.
Page fault handler swaps
in data
0xFFFF_FFFF
Physical Address Space
0x0000_0000
Physical RAM
0xFFFF_FFFF
Process B
Virtual Address Space
0x0000_0000
0xFFFF_FFFF