

# *ECE390*

## *Computer Engineering II*

### *Lecture 18*



Dr. Zbigniew Kalbarczyk  
University of Illinois at Urbana- Champaign

### *Outline*



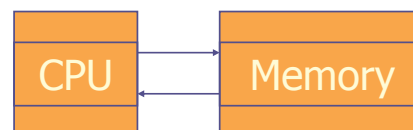
- Sound
  - Direct Memory Access (DMA)
  - Digital Signal Processing (DSP)

## Overview of DMA

- DMA transfers large blocks of data
- Reading sectors from hard drives
- Writing PCM data to the DSP

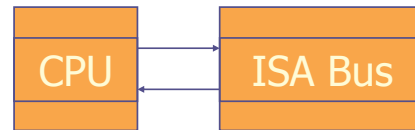
## Reading Memory

- CPU executes: `mov eax, [ds:esi]`
  - Provides address
  - Requests data
- Waits
  - Minimum of one bus clock cycle
- Memory returns data



## *Writing to a Device*

- CPU executes: `out dx, al`
- Data transfers across the ISA bus
  - 1 to 4 bus clock cycles
- Transfer complete

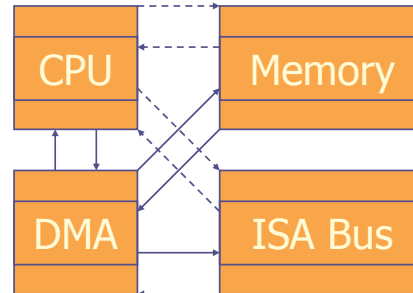


## *Waiting Is Slow*

- A 300MHz CPU on a 66MHz bus
  - 4.5 CPU cycles per bus cycle
  - 5 CPU cycles reading
  - 5 CPU cycles writing
- Wastes 8 cycles, or 80%

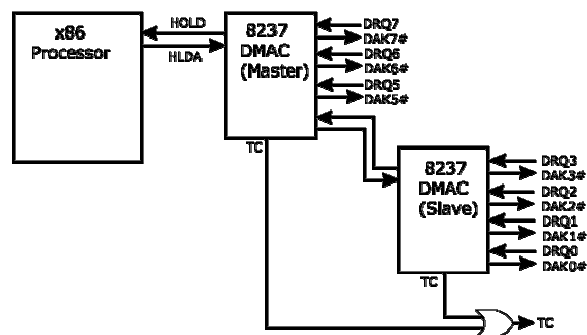
## Direct Memory Access

- CPU programs DMA and device
- CPU releases data bus
- DMA transfers data



## CPU ó DMA Interaction

- Master DMA asserts HOLD
- CPU acknowledges with HLDA



## *DMA Transfer Modes*

---

- DMA has four modes
  - **Single** – releases the HOLD after each byte of data is transferred
  - **Block** – automatically transfers the number of bytes indicated by the count register for the channel
  - **Demand** – transfers data until an external EOP (end of process) is input or DREQ input becomes inactive
  - **Cascade** – is used when more than one 8237 is present in the system
- Two options for each mode
  - Single cycle
  - Auto-initialized

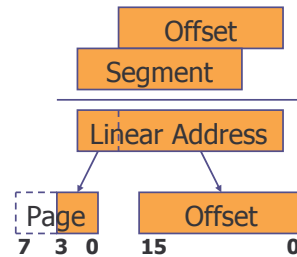
## *DMA Transfer Modes (cont)*

---

- Master DMAC and Slave DMAC
- Minimum of 1 byte
- Maximum of 64KB \* transfer size
- DMA can sustain max 4.166MB/s
  - 16bit 44KHz PCM data requires 88KB/s

## DMA Page Register

- DMA addresses memory by Page:Offset
  - 16 64KB pages
  - All below 1MB
  - Transfer buffer cannot cross page boundary

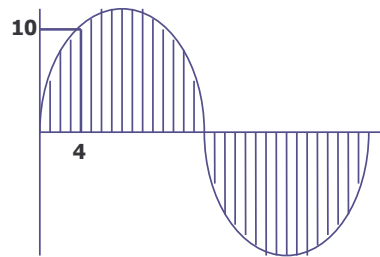


## Summary of DMA

- DMA can transfer large blocks of data
  - From memory to device
  - Or from device to memory
- Saves CPU time when accessing
  - Hard drives
  - Floppy drives
  - Soundcards

## Overview of the DSP

- Soundcards come in ISA, PCI, and onboard formats
- They play and record PCM sounds

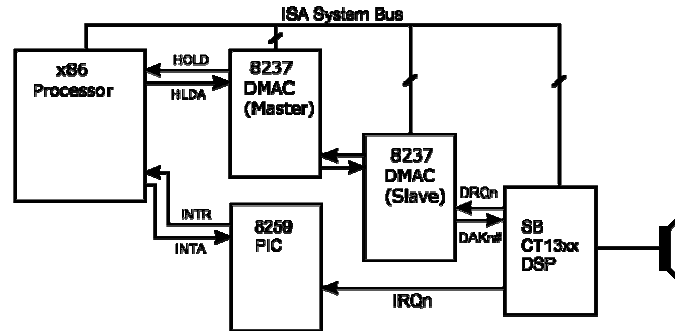


## Interface to the DSP

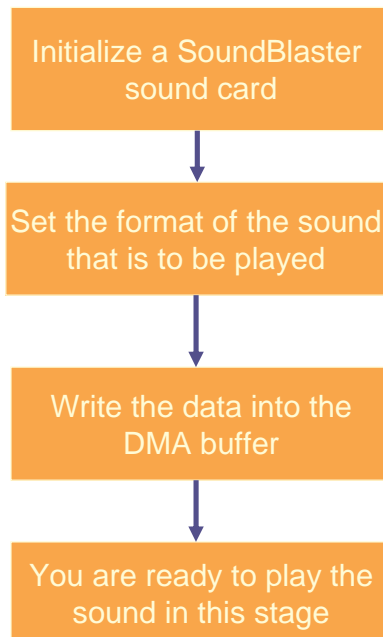
- By default, the 16bit DSP has
  - I/O Base 0220h
  - IRQ 5
  - DRQ 1
  - DRQ 5
- Held in **BLASTER** environment variable
  - `A220 I5 D1 H5`

## Programming the DSP

- Step 1: Reset the DSP
- Step 2: Set timing constant
- Step 3: Program a DMA transfer



## Playing Sound Setting up the Environment



## Playing Sound Example Code

```
invoke _SB16_Init, dword _SoundISR    ; initialize a soundblaster sound card

invoke _SB16_GetChannel
mov   [DMA_Channel], al

invoke _SB16_SetFormat, dword 8, dword 22050, dword 0

invoke _DMA_Allocate_Mem, dword DMA_BUFFERSIZE, dword DMA_Sel, dword DMA_Addr

invoke _DMA_Allocate_Mem

invoke _LockArea, word [DMA_Sel], dword 0, dword DMA_BUFFERSIZE

; suppose you [SoundOff] is the offset of your data of sound,
; and the size of your wave file
; is equal to DMA_BUFFERSIZE

; next, write the data of sound into DMA Buffer
add   dword [SoundOff], 44
mov   ecx, DMA_BUFFERSIZE
sub   ecx, 44

mov   es, [DMA_Sel]
mov   edi, 0
mov   esi, dword [SoundOff]
rep   movsb

; Now, you are ready to play sound
movzx eax, byte [DMA_Channel]
invoke _DMA_Start, eax, dword [DMA_Addr], dword DMA_BUFFERSIZE,
      dword 1, dword 1

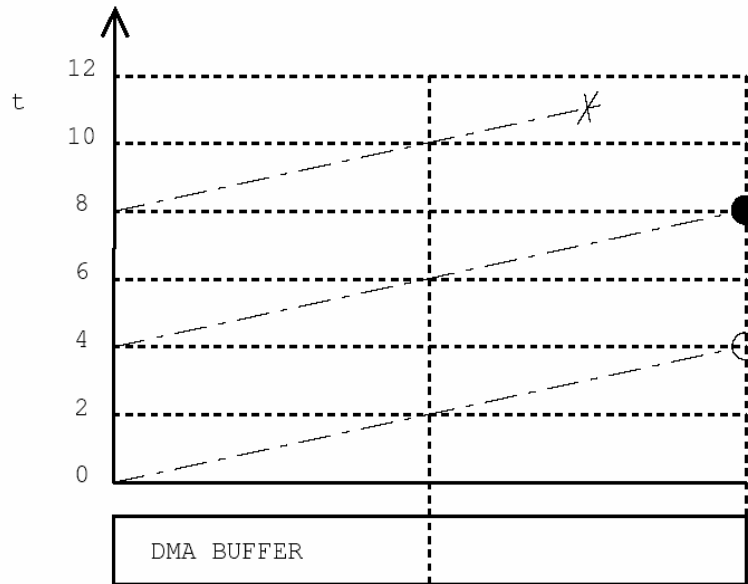
mov   eax, DMA_BUFFERSIZE
sub   eax, 44
Z.   invoke _SB16_Start, eax, dword 0, dword 1
```

17

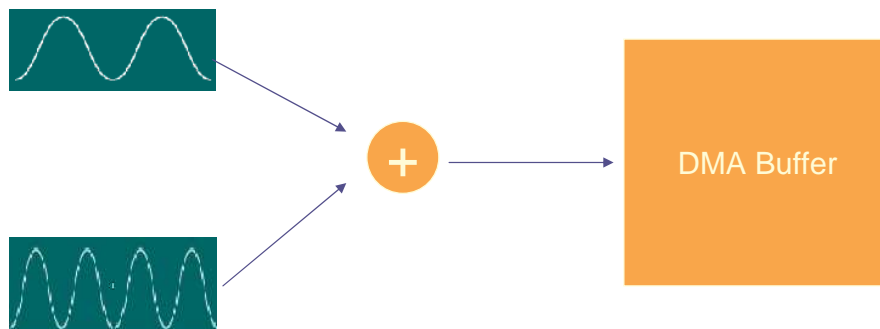
## How to play a long sound

- Maximum size of DMA buffer is 64KByte
- Usually, the size of the wave file is larger than that
- What can we do?

# Solution



# How to play 2 sounds at a time?



## *Audio Mixing*

- Two sound streams
- One speaker
- Average each sample together
  - MMX
  - Watch out for degenerate edge cases
- Lowers sound quality
  - Bad for 8bit; not so bad for 16bit

## *Appendix: Pmodelib Support*

- Initialization
  - DMA\_Alloc\_Mem (int Size, short\*Sel, long \*Address)
  - DMA\_Lock\_Mem ()
  - SB16\_Init (void(\*)() Callback)
  - SB16\_GetChannel ()
  - SB16\_SetFormat (int Bits, int Rate, bool Stereo)
  - SB16\_SetMixers (long Master, long Wav, ... (4 more) )
  - DMA\_Start (int Channel, long Address, long Size, bool AutoInit, bool Write)
  - SB16\_Start (long size, bool AutoInit, bool Write)
  - SB16\_Stop ()
  - DMA\_Stop (int Channel)
  - SB16\_Exit ()