Top Level Design

- OBJECTIVE: Learn How to Develop A Top Level Design
- OBJECTIVE: Learn Key Information Necessary For Top Level Design
- OBJECTIVE: Learn How to Document Top Level Design

Top Level Design Block Diagram

- The Main Purpose of the Top Level Design Is the Where
- Make a List of the What and Find a Home For Each Requirement.
  - » Can Use a Matrix
  - » Modularize As Much As Possible. We Will Define Functional Subsystems
- Define Interface Between the Subsystems
- After The Subsystems and Interfaces are Defined, We Will Develop The Derived Requirements.
  - » Assign Portions of Time Line, Size, Power etc to Each Subsystem

Make a List of the What and Find a Home For Each Requirement

Interfaces to Outside World {Parallel and Serial Ports}

We will Create an I/O Subsystem Module for Interfaces

CPU

We Will Create A CPU Subsystem (Brains of System)

Memory

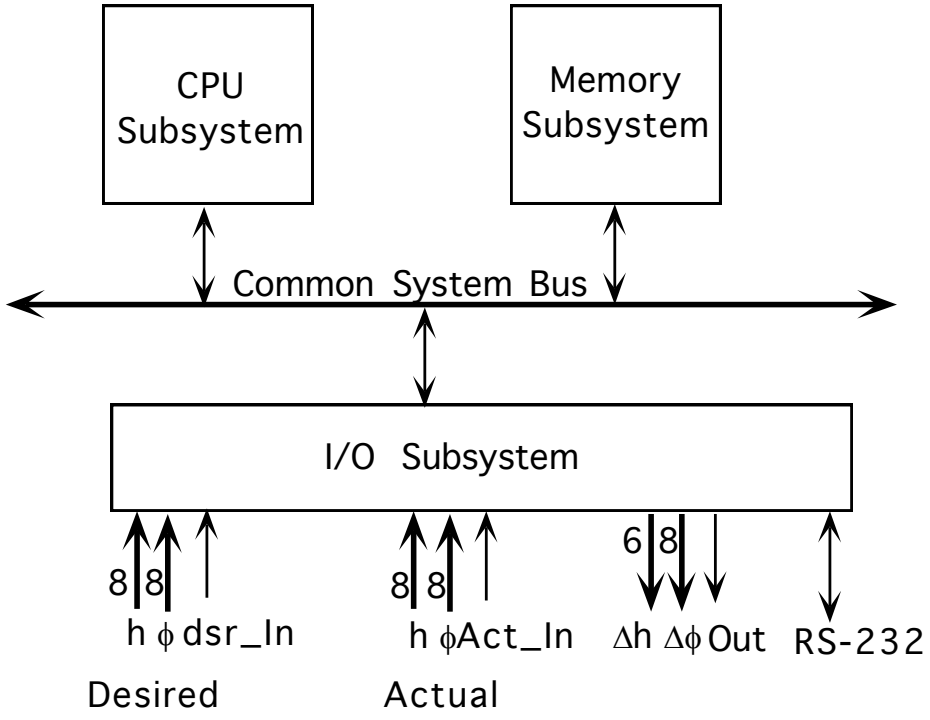
We Will Create a Memory Subsystem Module

Programs

Make Sure The Programs Are Accounted For

Memory Map and Allocations (Enough Memory For Programs,Data)

Subsystems on Common Bus Organization



Requirements/Subsystem Matrix

Requirements Matrix Maps Requirements Into Appropriate System

	CPU Subsystem	Memory Subsystem	I/O Subsystem
Requirement			
Current Position			2 8-bit Par. Inputs
Desired Position			2 8-bit Par. Inputs
Output Deltas			6,8-bit Par. Outputs
Arithmetic Ops	2's comp. Integer		
Data Requirement		2kbyte Min RAM	
Data Size	8 bit	8 bit	8 bit
Program		4kbyte Min ROM	
Debug Data		2kbyte Min RAM	
Debug Prog		4kbyte ROM	
Debug Input/Output			RS232
FunctionalUpdate	2,000 Instrs/Update		

Common Bus Organization

- CPU Provides All Addressing And Must Be Able To Access All Addresses/Data In System
- Address Map Is A Must
- “Which Came First, Chicken or Egg ?”
  - How Do We Get Data In/Out ?
  - What Does System Bus Look Like ?

Common Bus Definition

- The Bus Signals Are Generated From The CPU In Our System: We Must Know A Little About The CPU In Order To Define The Bus.
- Based On Requirements, The CPU Can Be Simple:
  - » Data 8 Bit Signed
  - » Addressing: < 1 Meg
  - » Integer Arithmetic
- These Requirements Easily Achievable With Wide Variety Of CPU's.
- We Can Base Our Decision:
  - » Prior Experience With CPU's (What You Are Familiar With)
  - » Development/Support Environment

Lets Choose The Intel 8086

Intel 8086 Signals

- Three Functional Busses:
  - » Address

A_{19} - AD_0

  - » Data

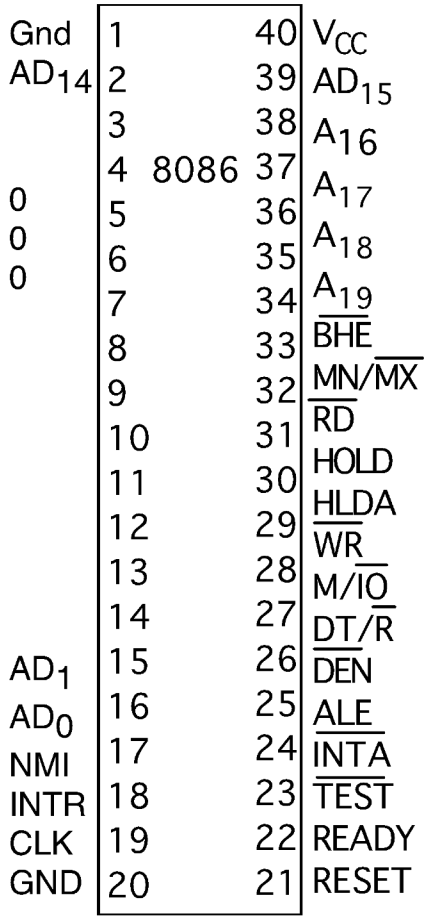
AD_{15} - AD_0

  - » Control

Group1 Data Xfer {ALE, RD, WR, BHE, M/IO}

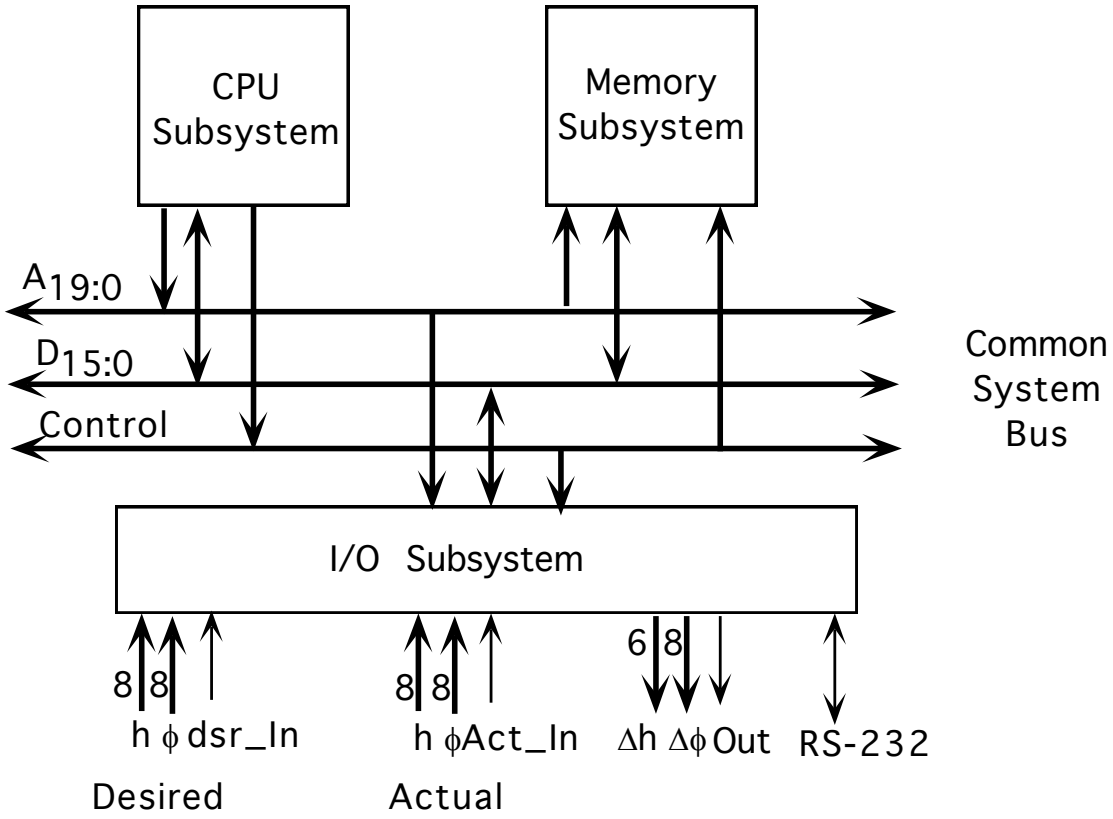
Group2 Interrupts {INTR,NMI,INTA}

Group3 Bus Control {HOLD,HLDA}



All Other Signals Will Stay Internal To CPU Subystem

Common Bus

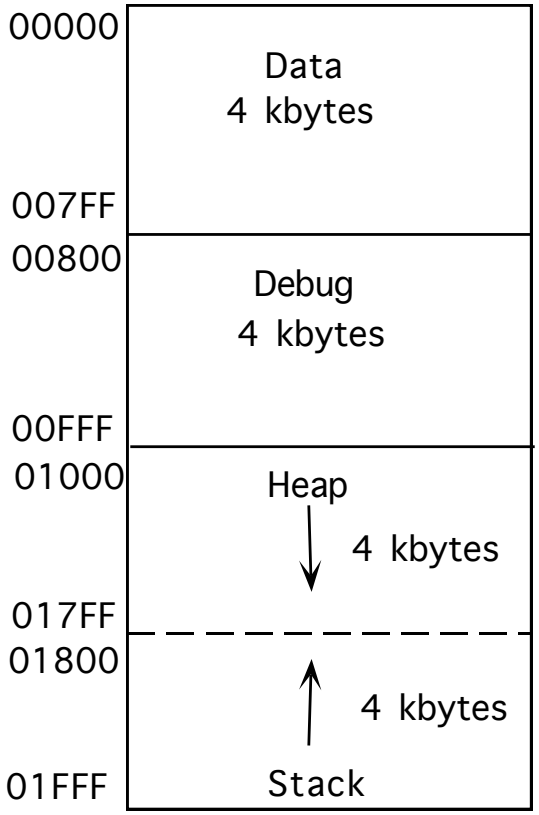

CPU Subsystem
Memory Subsystem
A19:0
D15:0
Control
Common System Bus
I/O Subsystem
8
8
h ϕ dsr_In
Desired
8
8
h ϕAct_In
Actual
6
8
Δh ΔϕOut
RS-232

Memory Map

- Purpose Of Memory Map Is To Show Design Teams Where They Should Decode Their Memory, Devices, Etc.
- Memory Must Be Large Enough To Hold:
  - » RAM Input Data, Temp Data, All Structures, Stack, Heap, etc...
  - » ROM Program, Initialization, Debug Monitor
- I/O Is Usually Memory Mapped (But Doesn't Need to Be)
  - » Provide A "Chunk" of Addresses (Assume Each Port Occupies an Address)

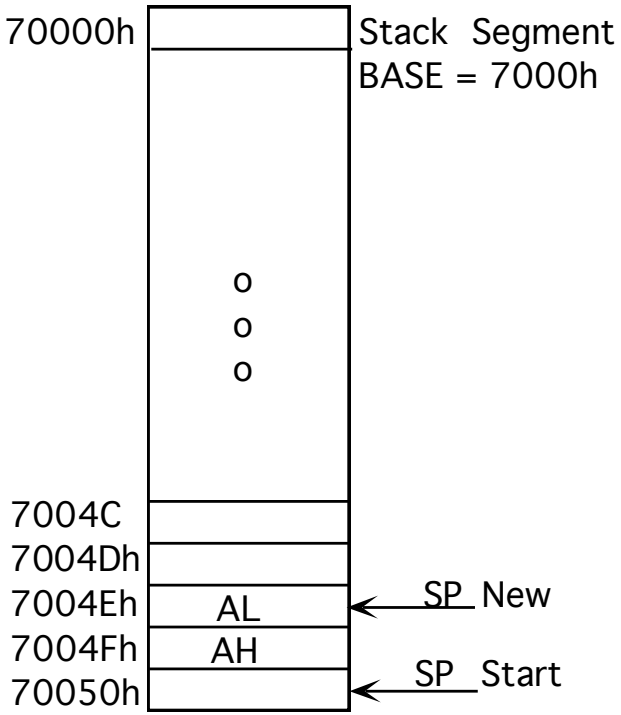
RAM Requirements

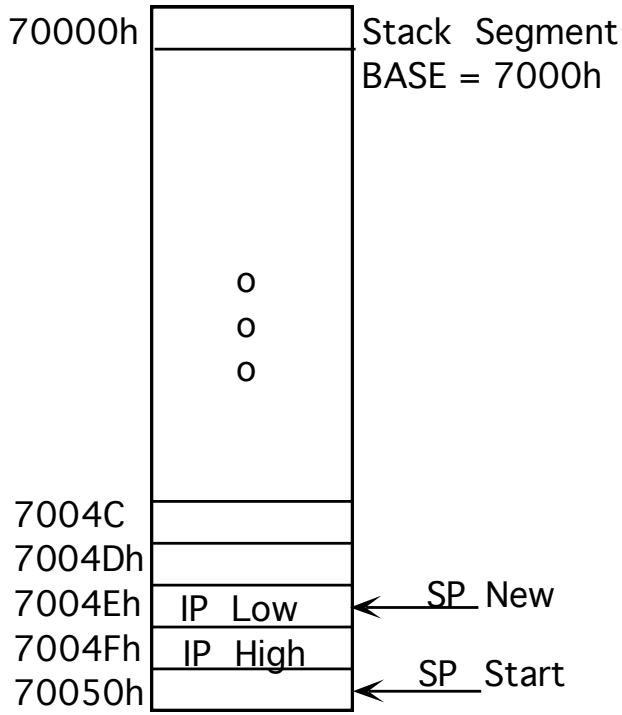
- Intel CPU's Want RAM Starting At Address x00000.
- We Need: Data Storage 2kbyte
  Stack 2kbytes
  Heap 2kbytes
  Debug 2kbytes
  8 kbytes
- Lets Double16 kbytes (All 4kbytes Segs)



Stacks

- Stack: Temporary Storage Space: (Points to Last Valid Entry)
  - » Pass Variables Between Subroutines Storage For Subroutine Calls
    - – PUSH AX -near CALL





Heaps

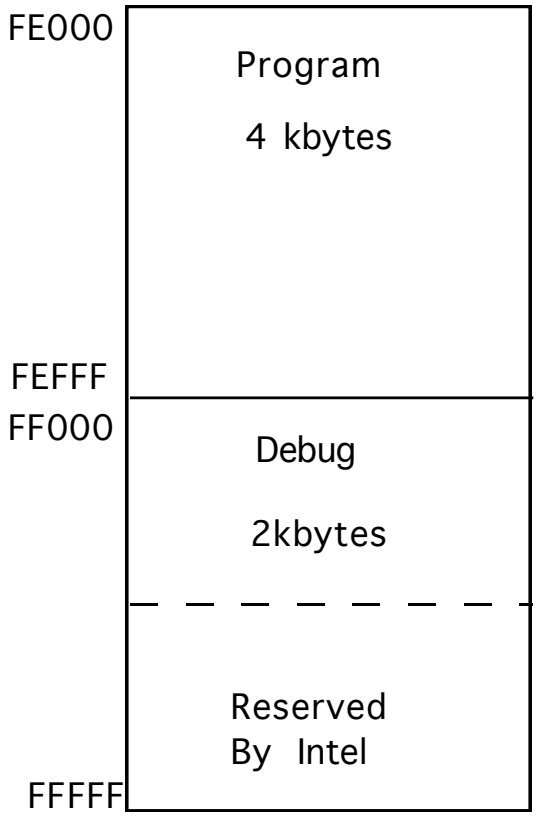
- Heap:
  - » Memory Allocation During Run Time
    - – Int a[];
    - – Size Not Defined at Compile
    - – a = Malloc(100*sizeof(int));
  - » OS Usually Takes Care of This. We Will Write Our Own

ROM Requirements

- ROM Needs To Hold:
  - » Program 4kbytes
  - » Debug 2 kbytes

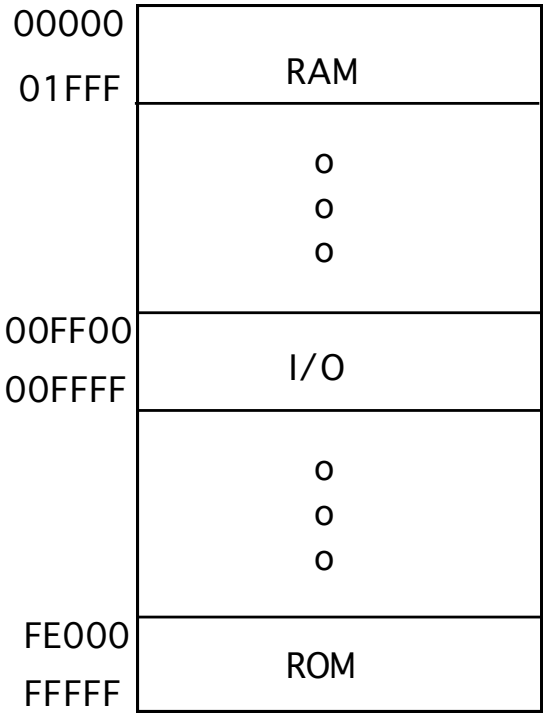
    Total 6 kbytes

  Go to 8 kbytes



Address Map

- Place Subsystems At Particular Address Block Range. This Guarantees That Parallel Design Teams Don't Overlap Address Decode.



Top Level Design Summary

- Create Subsystems
  - » Modularize the System
- Requirements Matrix
  - » Guarantee All Requirements Are Accounted For
- Define Internal Interfaces
  - » Define Signals/Data Structures Between Subsystems
- Define Memory Map
  - » Partitions System Address Space To Eliminate Overlap

Allows Multiple Groups To Work Simultaneously