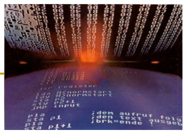


Chapter 2
8086 Programming Model

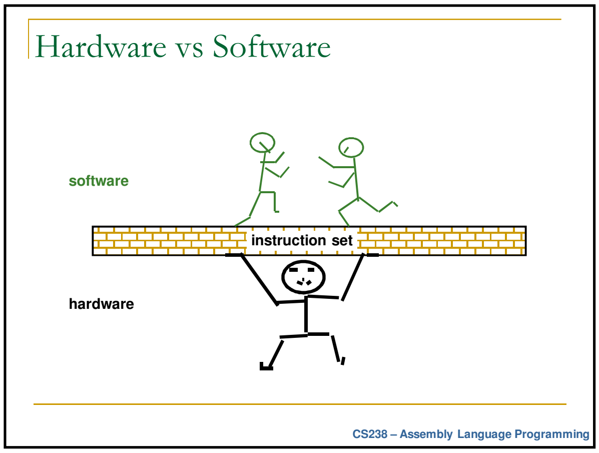
CS238 – Assembly Language Programming
Amarnath Jasti

Review

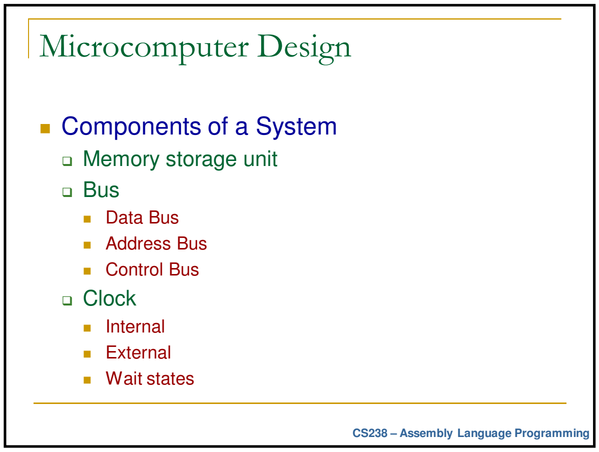
- Where assembly is used?
  - Embedded Systems
  - Perform specific h/w functions
  - Drivers
  - Flexibility (Ex: Programming in robotics)
- Computer system
  - Hardware (CPU, PPD, Memory, etc)
  - Software (OS, Drivers, Application software, etc)

Assembly Language

- Assembly Language definition:
  - Machine specific programming language with a one-to-one correspondence between the statements and the computers native machine language and is specific to the processor or processor family
  - Note: Instructions in assembly are designed to match a computers machine instruction set and hardware architecture.

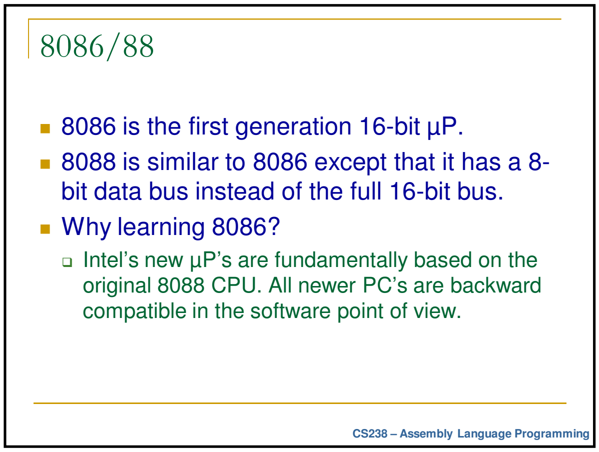


Hardware vs Software



Microcomputer Design

- Components of a System
  - Memory storage unit
  - Bus
    - Data Bus
    - Address Bus
    - Control Bus
  - Clock
    - Internal
    - External
    - Wait states



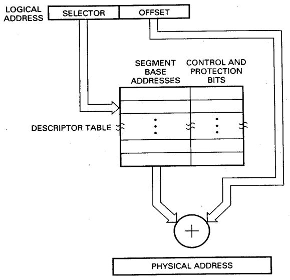
8086/88

- 8086 is the first generation 16-bit µP.
- 8088 is similar to 8086 except that it has a 8-bit data bus instead of the full 16-bit bus.
- Why learning 8086?
  - Intel’s new µP’s are fundamentally based on the original 8088 CPU. All newer PC’s are backward compatible in the software point of view.

8086/88

- 20-bit address bus = 1MB (2^{20} =1048576 bytes)
- I/O mapped or Isolated I/O addressing in which I/O ports are addressed individually. 8088 supports up to 256 I/O ports.
- Memory Map – 00000h → FFFFFh
  - In practice memory is divided into 64kb blocks or pages, i.e. blocks from 0000h → FFFFh
  - Physical address = (Segment Address * 16d) + Offset

Physical Address



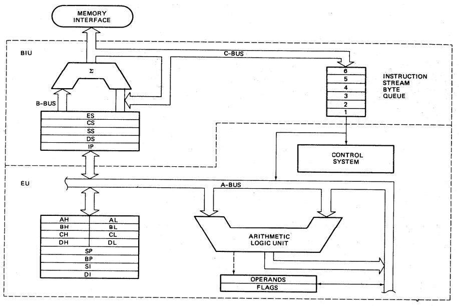
Examples

- 0000h:0000h
- 1000h:0050h
- 0800h:8005h
- 0900h:7005h
- F000h:FFFFh

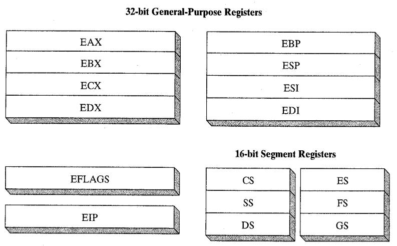
Memory Map

- Question:
  - Given a ROM size of 256KB starting at A0000h, determine the ending address of ROM?
    - 256KB = 256 * 1024 = 262144 bytes = 40000h -1 = 3FFFFh
    - So ending address = A0000h + 3FFFFh = DFFFFh

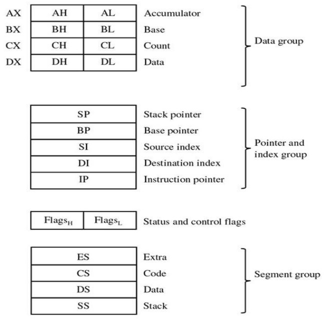
8086 Block Diagram



Register set



8086/88 Registers



AX AH AL Accumulator
BX BH BL Base
CX CH CL Count
DX DH DL Data
Data group

SP Stack pointer
BP Base pointer
SI Source index
DI Destination index
IP Instruction pointer
Pointer and index group

$Flags_H$ $Flags_L$ Status and control flags

ES Extra
CS Code
DS Data
SS Stack
Segment group

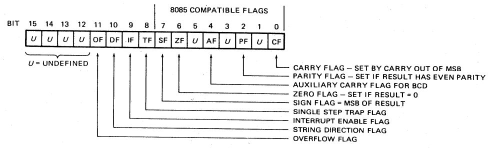
Data Registers (AX, BX, CX, DX)

- General Purpose Registers – arithmetic calculations, temporary data storage, data transfer.
- AX (AH – AL) – Accumulator
- BX (BH – BL) – Base Register; holds blocks base address.
- CX (CH – CL) – Count register
- DX (DH – DL) – Data or Destination Register (Used by I/0)

8086/88 Registers

- Instruction Pointer (IP)
  - 16-bit register
  - Maximum addressing space in KB?
  - Used in conjunction with CS to generate 20-bit physical address (CS:IP)
- Flag register

8086 Flags


8085 COMPATIBLE FLAGS
BIT
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
U
U
U
U
OF
DF
IF
TF
SF
ZF
U
AF
U
PF
U
CF
U = UNDEFINED
CARRY FLAG – SET BY CARRY OUT OF MSB
PARITY FLAG – SET IF RESULT HAS EVEN PARITY
AUXILIARY CARRY FLAG FOR BCD
ZERO FLAG – SET IF RESULT = 0
SIGN FLAG = MSB OF RESULT
SINGLE STEP TRAP FLAG
INTERRUPT ENABLE FLAG
STRING DIRECTION FLAG
OVERFLOW FLAG

8086/88 Registers

- Segment Registers
  - These registers are required to address total RAM space(1 MB) where as all pointer registers are only 16-bits
  - CS – code segment: Program instructions currently being executed are located.
  - DS – Data Segment – Program data is located
  - ES – Extra segment – used to define another segment in addition to data segment to address data beyond 64KB.
    - NOTE: Address (segment) values cannot be loaded directly into segment registers. Data registers need to be used as intermediate registers for data transfer.
    - EX:
      - MOV AX, 0B800h
      - MOV ES, AX

8086/88 Registers

- Stack Pointer – always points to the current location of the 'Top of the Stack'
  - It is used in conjunction with SS to get 20-bit address (SS:SP)
- Base pointer (BP) → (DS: BP)
- Source Index register (SI): Used to point to a byte or word in the current data segment that needs to be fetched as a part of a block of data. (DS:SI)
- Destination Index Register (DI) – similar to SI but used as destination address in ES for data transfer (ES:DI).

Memory Map - RAM

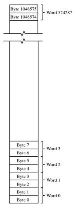
- Interrupt Vector Table
  - Table containing addresses of interrupt service routines.
- BIOS data
  - Temporary storage used by BIOS.
- DOS data
  - Area of RAM used to store temporary program data such as current file name, directory name etc.
- Resident DOS
  - Main DOS program code
- User RAM Space
  - Available for users for program code
- Video RAM
  - Video RAM is physically located on the video adaptor but must be mapped into the primary memory address space.

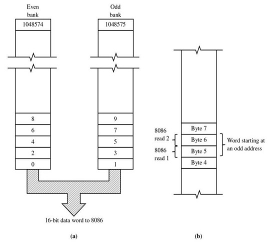
Memory Map - ROM

- Extension ROM space
  - Reserved for additional hardware device ROM code
  - Provides space where routines can be supplied to initialize and control 'unknown' hardware.
- User ROM space
  - Provided for users to install their own program code on ROM
- BASIC ROM Space
- BIOS ROM Space

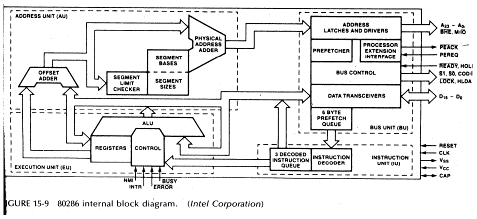
8086 Memory organization

- Segmented addressing (cf. linear addressing)
- Even and Odd memory banks





80286 (32 bit) Block Diagram



IGURE 15-9 80286 internal block diagram. (*Intel Corporation*)

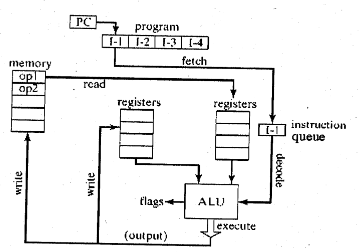
Review

- What bit general purpose register is BX?
  - 16
- Which one of them is not a segment register?
  - CS, ES, DS, GS, SS, FS.
    - All are segment registers
- Define IP?
  - IP points to the address where next instruction that needs to be executed is present.

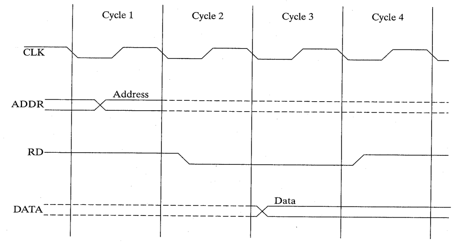
Memory Management

- Real-address mode
  - 1MB of memory can be addressed (00000–FFFFF)
  - One program at a time with interrupts
  - Standard memory restrictions
  - Ex: MS-DOS
- Protected mode
  - Multiple programs with reserved memory
  - Ex: Windows and Linux
- Virtual-8086 mode
  - Protected mode
  - Simulates 8086 environment with 1MB dedicated memory
  - Hardware/software restrictions apply

Instruction Execution Cycle



Machine Read cycle

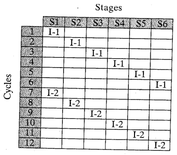


Fetch and Execute

- PC – Program counter
- Instruction queue
- Fetch and Execute
  - Fetch
  - Decode
  - Fetch operands (Memory Read)
  - Execute
  - Store O/P operand (Memory)
- Multi tasking
  - Scheduler (Time: slice)
  - Round-robin scheduling
  - Task switching or swapping

Multi-Stage Pipelining

- Stages
  1. Bus interface Unit (BIU)
  2. Code Prefetch Unit
  3. Instruction Decode Unit
  4. Execution Unit
  5. Segment Unit
  6. Paging Unit

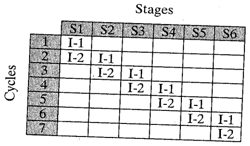


Stages

Cycles	S1	S2	S3	S4	S5	S6
1	I-1					
2		I-1				
3			I-1			
4				I-1		
5					I-1	
6						I-1
7	I-2					
8		I-2				
9			I-2			
10				I-2		
11					I-2	
12						I-2

- For *k* execution stages, *n* instructions require *(n*k)* cycles to process.

Six stage pipelined Instruction Execution

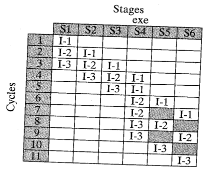


Stages

Cycles	S1	S2	S3	S4	S5	S6
1	I-1					
2	I-2	I-1				
3		I-2	I-1			
4			I-2	I-1		
5				I-2	I-1	
6					I-2	I-1
7						I-2

■For *k* execution stages, *n* instructions require *(k+(n-1))* cycles to process.

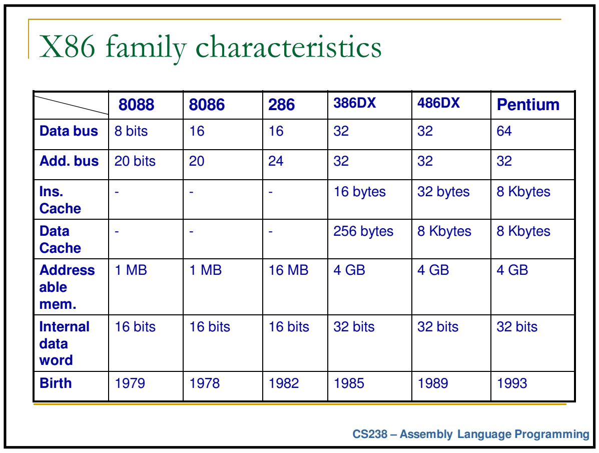
Superscalar Pipelined execution



Stages

Cycles	S1	S2	S3	S4 exe	S5	S6
1	I-1					
2	I-2	I-1				
3	I-3	I-2	I-1			
4		I-3	I-2	I-1		
5			I-3	I-1		
6				I-2	I-1	
7				I-2	▨	I-1
8				I-3	I-2	▨
9				I-3	▨	I-2
10					I-3	▨
11						I-3

•Using single pipeline, for *k* execution stages (where one stage takes two clock cycles), *n* instructions require *(k+2n-1)* cycles to process.
•Superscalar 6-stage pipeline: S4 is divided into two pipelines. Takes *(k+n)* cycles



X86 family characteristics

	8088	8086	286	386DX	486DX	Pentium
Data bus	8 bits	16	16	32	32	64
Add. bus	20 bits	20	24	32	32	32
Ins. Cache	-	-	-	16 bytes	32 bytes	8 Kbytes
Data Cache	-	-	-	256 bytes	8 Kbytes	8 Kbytes
Address able mem.	1 MB	1 MB	16 MB	4 GB	4 GB	4 GB
Internal data word	16 bits	16 bits	16 bits	32 bits	32 bits	32 bits
Birth	1979	1978	1982	1985	1989	1993

CS238 – Assembly Language Programming