Experiment 6

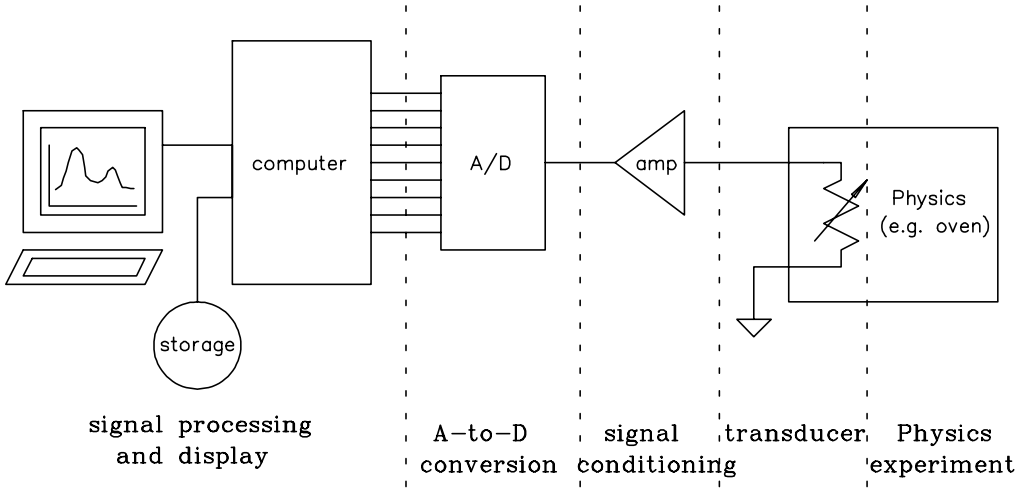
An introduction to using microcomputers in a physics laboratory

To use microcomputers as laboratory instruments, we need to learn the basics of their architecture, and the ways to control them at the level of electrical signals. We "look under the hood" of a computer-based data acquisition system and learn how to control its hardware through low-level programming.

Introduction

Hardware

- microcomputers as lab instruments



- inside an IBM PC
  - the mother board: CPU, RAM, BIOS, some I/O & timing circuits
  - the I/O bus: expansion cards
    Note: CPU often has a separate, faster & wider bus to access memory (RAM)
  - I/O card and the connection to the outside world
  - operation of the computer boils down to controlled transfer and manipulation of bits (digital on/off, 0V/5V levels) among various components

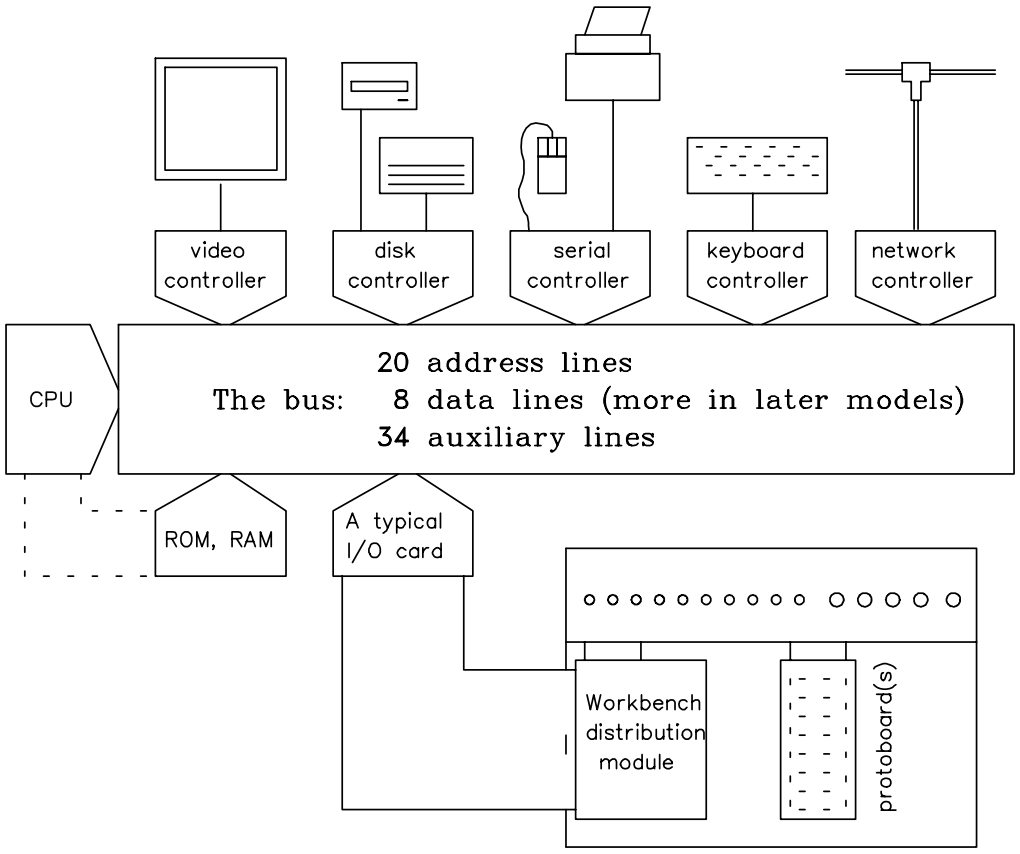


Figure 6.1: Essential elements of a personal computer

  - complex operations: many clock cycles; some devices may have to wait; computer is not a "real-time" device! — but latency is small
  - structure of a computer bus: *address lines* — which device is getting/sending data; *auxiliary* — the timing of the transfer, handshaking; *data lines* AD0–AD7 — the data may be a real number, a machine instruction, a part of an address, *etc.*

- A typical I/O interface:
  - I/O card inside a PC
  - ribbon cables
  - distribution module at the workstation
  - optoisolators and separate commons
  - 4 x 8-bit input & 4 x 8-bit output parts
  - multiplexed 4-input, 12-bit A/D converter (bipolar ± 10V)
  - 2 x 12-bit D/A outputs (bipolar ± 10V)
  - counter & timer circuits

Software

- Programming languages: see the summary in Table 6.1

Table 6.1: The hierarchy of programming languages

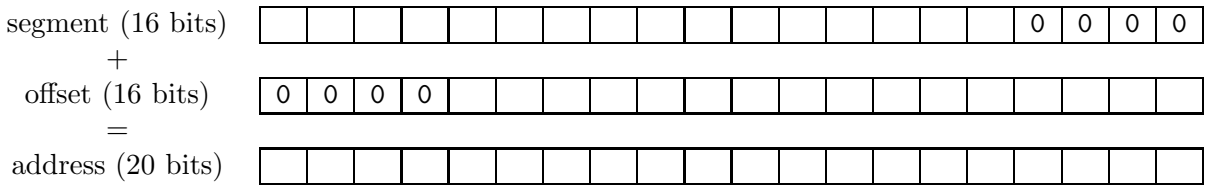
Language	Examples	Comments	C†
4th generation	code generators, AI	large projects, productivity	
High-level	Fortran, Pascal, Basic	easy on humans, hw independent	•
Low-level	Assembler (mnemonics)	hard, but full control of hw	•
Machine	operation codes	"understood" by the CPU	
Hardware	bits, 0/5V voltage levels		

† C bridges the two layers; is a high-level language but allows low-level extensions.

- Compilers: computer programs that translate the human-readable source code into machine-executable op-codes.
- Op-codes: for example, about 300 in six functional groups for 8088/8086:

data transfer	`mov, in, out, xchg`
arithmetic	`add, sub, inc, dec, mul, div`
logical	`and, or, xor, not`
string manipulation	
control transfer	`jmp, call`
processor control	

- 8088/8086 registers: Table 6.2 shows a list of 14 internal (to CPU) memory locations with very fast access used to keep track of where in the program the CPU is. Each register is 16-bit wide. However, General Registers (see Table 6.2) can also be addressed in two 8-bit chunks (`xL` and `xH` for Low and High bits, respectively).
- SEGMENT:OFFSET addressing: 16 bits can address 65,536 locations = size of segment; 20 bits can address 1MB



- IBM PC memory map is summarized in Table 6.3

This is just an example of a particular architecture (IBM PC); other microprocessors and micro-controllers differ in the details, but are fundamentally similar to the above description.
More to come...

Table 6.2: 8086/8088 CPU registers

16-bit segment	high 8 bits	low 8 bits	comments
General registers:			
AX	AH	AL	accumulator register
BX	BH	BL	
CX	CH	CL	
DX	DH	DL	
Addressing registers:			
SI	–	–	
DI	–	–	
BP	–	–	
Control registers:			
SP	–	–	
IP	–	–	
Flags†	–	–	
Segment registers:			
CS	–	–	Code Segment pointer
DS	–	–	Data Segment pointer
ES	–	–	
SS	–	–	

† Flags: consists of individual bits which flag certain conditions.
E.g., bit `ZF=on` (Zero Flag) means the last instruction yielded a 0 as a result.

Table 6.3: IBM PC memory map

`00000 - 003FF`	I/O
`00400 - 0047F`	BIOS Ram
`00480 - 005FF`	RAM for special purposes (ROM Basic)
`00600 - 9FFFF`	Program memory
`A0000 - AFFFF`	VGA video expansion
`B0000 - B0FFF`	IBM monochome video
`B8000 - BFFFF`	CGA video memory
`C0000 - CFFFF`	reserved for video expansion
`D0000 - D7FFF`	ROM, usually not all installed
`D8000 - DFFFF`	”
`E0000 - E7FFF`	”
`E8000 - EFFFF`	”
`F0000 - F3FFF`	reserved for ROM
`F4000 - F5FFF`	”
`F6000 - FDFFF`	Basic in ROM in the original IBM PC
`FE000 - FFFFF`	BIOS ROM