Wrap-up

Two different types of microprocessor-based systems:

- general-purpose
- embedded: dedicated to a specific task as part of a larger product

Those of you who work in this area are much more likely to end up in the latter field. Contrast ~80 million PCs, few million servers/mainframes etc. with over 3 billion embedded systems per year (1995 data).

Categories of embedded systems:

- general: thinly veiled, somewhat specialized computers
  - video game console
  - WebTV box
  - info kiosk
- control: feedback control of physical systems
  - vehicle engines
  - manufacturing (milling machines, assembly line)
  - aircraft flight control
- signal processing: high-bandwidth data stream manipulation
  - filtering, analysis, compression, decompression
  - radar, modems, medical imaging,
- communications/networking
  - telephone switches, network routers, voice-mail systems

Many systems will have incorporate facets of more than one category.

General-purpose systems:

- performance, cost are primary goals
- often shoot for max performance at given price point
- processor (and often peripheral) choices dictated by software compatibility

Embedded systems:

- cost is usually paramount (even for NASA)
- must meet functional requirements at minimum cost
- must consider system cost:
  - CPU, memory, I/O devices
    - integrated microcontroller is usu. best bet if capable enough
  - circuit board size (smaller is cheaper)
  - connectors
  - power supply (# of batteries, AC adapter req'd?)
  - software development cost (non-recurring engineering (NRE), but may be largest single item)
  - life-cycle issues:
    - manufacturing cost
    - testability
    - maintenance: availability of spare parts, ease of repair/upgrade

functional requirements very diverse, application-specific

1. performance: application defines "fast enough", going significantly faster is a waste of money/power
   - often real time: must respond by fixed deadline
     - hard real-time: response is useless if late (elevator control)
     - soft real-time: response less useful if late (video display quality degrades)
   - must guarantee response in the face of:
     - interrupts (mult priorities)
     - other real-time tasks
     - DRAM refresh
     - cache misses (often don't use caches)
     - worst-case combination of above
   - peak, even average performance a non-issue

2. size
   - fit in shirt pocket
   - fit into left-over space (e.g., digital camera)

3. weight
   - portability
   - transportation: weight costs money
     - cars, airplanes, satellites

4. power
   - run on battery for hours, months, years
     - larger battery may violate weight constraint
   - standby vs. active power (cars)
   - cooling constraints on line-powered devices (no fan)

5. safety/reliability
   - consequences of failure
   - failure modes: fail-stop, random output
     - watchdog reset, power cycle, known good state
   - electro-mechanical interlocks, backup system
   - more extreme: redundancy (SW & HW), voting circuits, etc.

6. time to market
   - better device that's late may not sell

7. environment
   - heat, vibration, shock (drop)
   - RF interference (pacemakers vs. microwaves)
   - lightning strikes (phone equipment)
   - water

Kicker: you won't be given a (complete) list of requirements up front; a big part of the job is usually to interpret/anticipate a customer's implicit desires to generate useful requirements.

Summary: successful microprocessor-based system design goes far beyond what you've learned here:

- optimization under complex, interdependent constraints

- digital design, software, analog design, electromechanics, control theory, signal processing

- work on a team with experts from other domains: manufacturing, marketing, human interfaces, etc.

- communication skills: explain your design, convince your boss/customer you did a good job, motivate colleagues

