Recap of last class

- Compilation optimization
  - Optimizing expressions
  - Optimize loops
  - Optimize function calls
  - Use registers efficiently

ECE 1175
Embedded Systems Design

Program Optimization II

Wei Gao

Program Optimization for Embedded Systems

- 1. Optimizing for execution time.
- 2. Optimizing for energy/power.
- 3. Optimizing for program size.
- Those goals may conflict with each other!

Execution Time Analysis

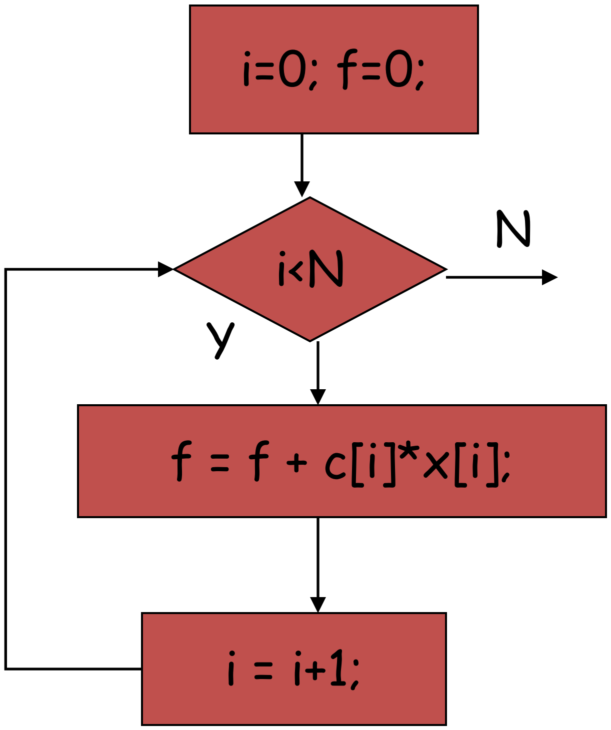
- Real-time embedded systems must meet deadlines
  - Predictability is the key
  - Inaccuracy: cache, pipeline, various optimizations
- Execution time analysis
  - Average-case
    - For typical data values, whatever they are.
    - Good for soft real-time systems
  - Worst-case
    - For any possible input set
    - Hard real-time
- Techniques for improving execution time?

Execution Time

- Affected by program path and instruction timing
- Program path depends on input data
  - Sensor readings, User input
- Instruction timing depends on
  - Cache behavior: memory access is slower
  - Instruction level variations
    - Floating point vs. integer operations

Program Path

```
for (i=0, f=0; i<N; i++)
   f = f + c[i]*x[i];
```



- Loop initiation block executed once.
- Loop test executed N+1 times.
- Loop body and variable update executed N times.

- Find the longest path length for execution time analysis.

Measurement-Driven Analysis

- CPU simulator.
  - I/O may be hard.
  - May not be totally accurate.
- Time stamping
  - Requires instrumented program.
  - Timer granularity
    - **`Gettimeofday`** on UNIX/Linux: **10 ms**
    - **`Gethrtime`** on Pentium: read a 64 bit clock cycle counter. and return the number of clock cycles since the CPU was powered up or reset: nanoseconds resolution.

2. Optimizing for Energy/Power

- Important for battery-powered systems and for system reliability and cost
- Energy: ability to do work.
  - Most important in battery-powered systems.
- Power: energy per unit time.
  - Important even in wall-plug systems---power becomes heat.
- We have classes on power management

Measuring Energy Consumption

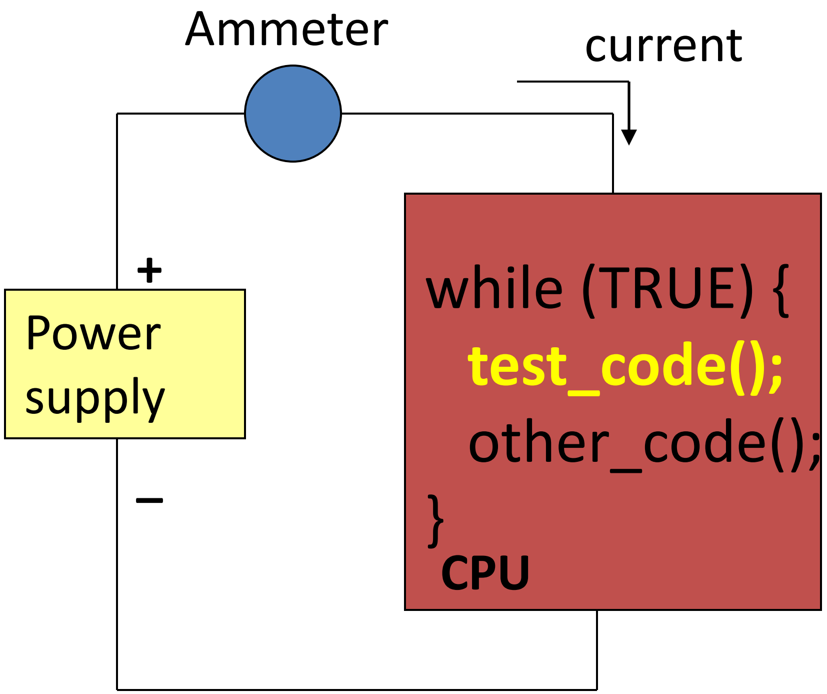
- Built-in system function calls
  - Example:
    - Android energy profiler
  - Inaccurate: the measuring function call consumes power, too.

  - Similar story: measuring app’s execution time
    - Example: Android TimingLogger

Measuring Energy Consumption

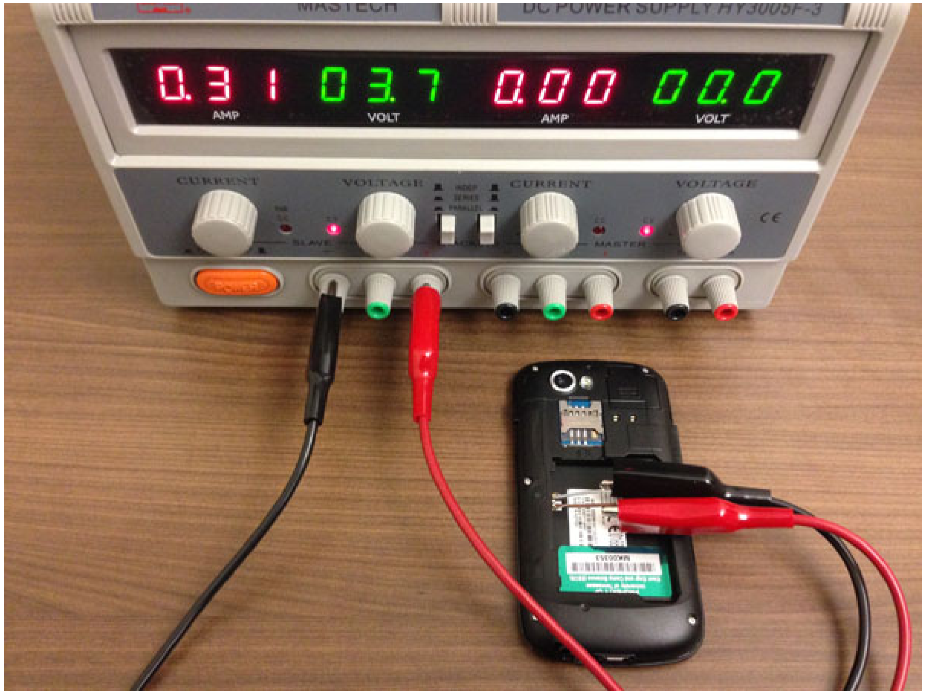
- Measure the power consumption for an instruction or a small code segment

1. Executes the code under test over and over in a loop
2. Measure the current flowing to the CPU
3. Delete the test code from the loop
4. Measure the current flowing to the CPU again
5. Calculate the difference



Measuring Energy Consumption

- Example: measuring the power consumption of a smartphone



Execution Time vs. Power Consumption

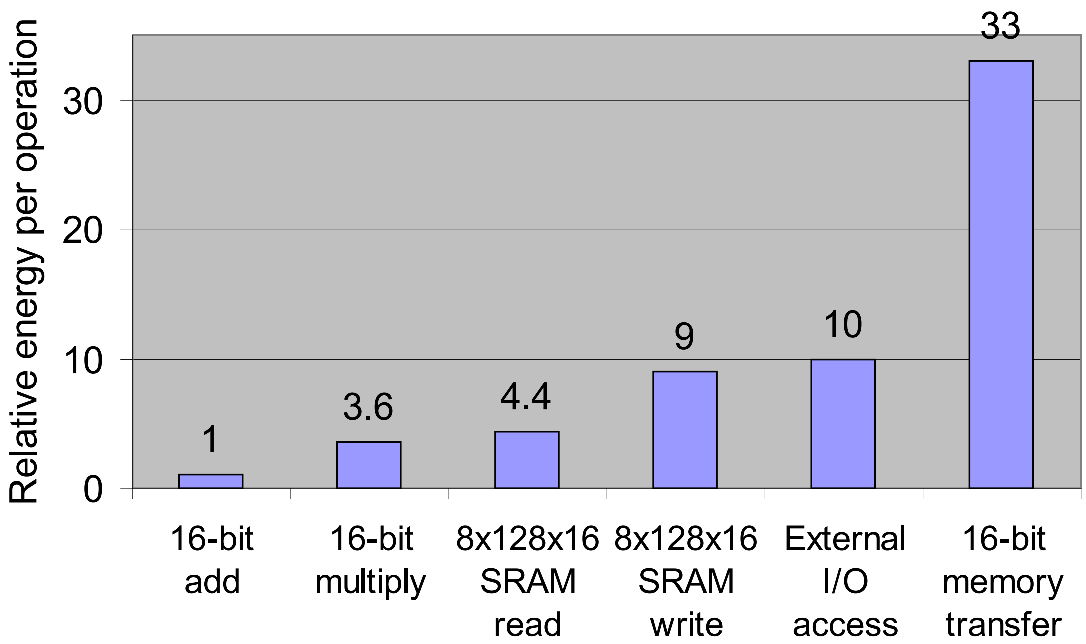
- Computation: execution time is proportional to power consumption
  - $O(n^3)>O(n^2)>O(n)>O(\log n)$
- However…
  - Don't forget the coefficient: $O(n^2) \sim k_1*n^2+k_2*n+k_3$
  - Time complexity vs. space complexity

	Time Complexity	Space Complexity
QuickSort	$O(n \log(n))$	$O(n)$
MergeSort	$O(n \log(n))$	$O(n)$
BubbleSort	$O(n^2)$	$O(1)$

  - Power consumed other than computation

Sources of Energy Consumption

- Relative energy of CPU per operation (Catthoor et al):



- Memory transfer is the most expensive operation
  - Biggest energy optimization comes from properly organizing memory
- Energy consumption: memory > caches > registers

How to optimize your program?

- Computation
  - Minimize the time complexity
  - Tradeoff between time complexity and space complexity
- Memory operation
  - Optimizing the cache use
  - De-segmentation
- I/O operations
  - Try to cluster data reads/writes
  - Minimize the number of device wakeups
  - Use buffer wisely!

3. Optimizing for Program Size

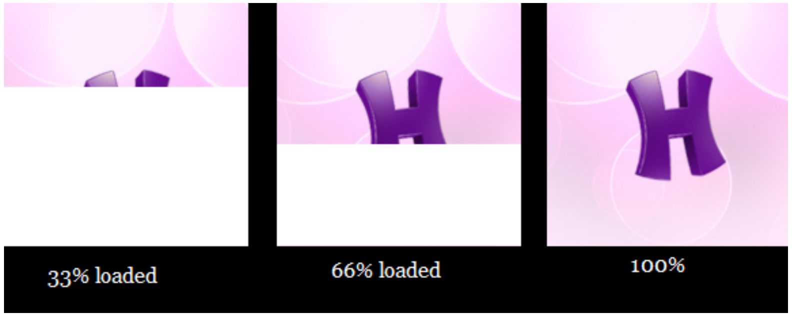
- Benefits
  - Reduce hardware cost;
  - Reduce power consumption.
- Size is determined by data and instructions
- Two opportunities:
  - Data;
  - Instructions.

Reduce Data Size

- Reuse constants, variables, data buffers in different parts of code.
  - E.x., pack multiple flags in one byte
    - int flag1=1; flag2=0; flag3=1 → one-byte flag: 101
  - Requires careful verification of correctness.
- Generate data using instructions
  - Instead of using static data with initial values
- Data compression
  - Tradeoff with computation complexity

Tricks with Data Compression

- Progressive loading
  - Utilizing "user experience"



- Adaptation to user need

Reduce Code Size

- Avoid loop unrolling.
  - Reduces loop overhead but increase code size
- Inlining?
  - Size of function
  - Number of calls
- Choose CPU with compact instructions.
  - Ex. DSPs (CISC)
- Some CPUs support dense instruction set
  - ARM Thumb, MIPS-16

Summary

- Basic compilation optimization
  - Expression simplification
  - Dead code elimination
  - Function inlining
  - Loop optimizations
  - Register allocation
- Optimization for embedded systems
  - Optimizing for execution time
    - Execution time analysis: Program path, instruction timing
    - Execution time metrics: Average-case, worst-case
    - Execution time measurement: trace analysis
  - Optimizing for energy/power
    - Measurement, sources of energy consumption, cache
  - Optimizing for program size
    - Reduce data size and code size