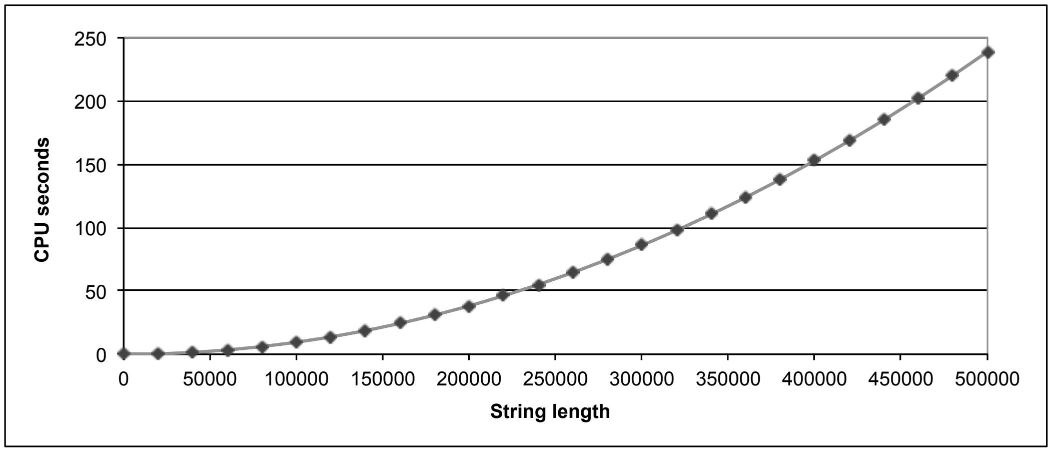
Example: lower

```
void lower(char* s) {
  size_t i;
  for (i = 0; i < strlen(s); i++)
    if (s[i] >= 'A' && s[i] <= 'Z')
      s[i] -= ('A' - 'a');
}
```

Optimization Blocker?



```
void lower(char* s) {
  size_t i;
  for (i = 0; i < strlen(s); i++)
    if (s[i] >= 'A' && s[i] <= 'Z')
      s[i] -= ('A' - 'a');
}
```

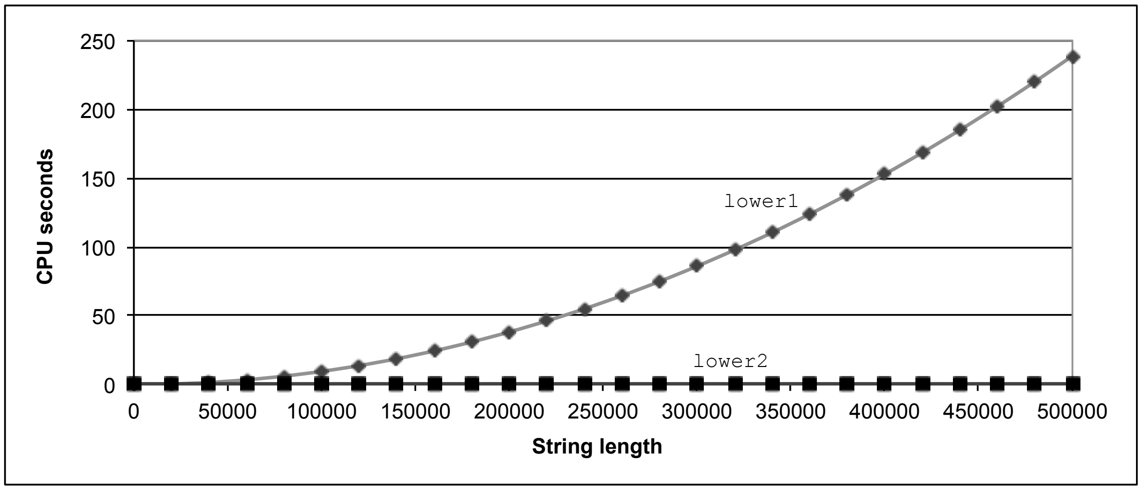
Lower Goto Form

```
void lower(char* s) {
   size_t i = 0;
   if (i >= strlen(s))
     goto done;
 loop:
   if (s[i] >= 'A' && s[i] <= 'Z')
       s[i] -= ('A' - 'a');
   i++;
   if (i < strlen(s))
     goto loop;
 done:
}
```

Improving Lower

```
void lower(char* s) {
  size_t i;
  size_t len = strlen(s);
  for (i = 0; i < len; i++)
    if (s[i] >= 'A' && s[i] <= 'Z')
      s[i] -= ('A' - 'a');
}
```

Improving Lower



Example: Memory Matters

```
/* Sum rows of n X n matrix a
   and store in vector b  */
void sum_rows1(double* a, double* b, long n) {
    long i, j;
    for (i = 0; i < n; i++) {
        b[i] = 0;
        for (j = 0; j < n; j++)
            b[i] += a[i*n + j];
    }
}
```

```
# sum_rows1 inner loop
.L4:
        movsd   (%rsi,%rax,8), %xmm0        # FP load
        addsd   (%rdi), %xmm0               # FP add
        movsd   %xmm0, (%rsi,%rax,8)        # FP store
        addq    $8, %rdi
        cmpq    %rcx, %rdi
        jne     .L4
```

Memory Aliasing (1)

```
/* Sum rows of n X n matrix a
   and store in vector b  */
void sum_rows1(double* a, double* b, long n) {
    long i, j;
    for (i = 0; i < n; i++) {
         b[i] = 0;
         for (j = 0; j < n; j++)
             b[i] += a[i*n + j];
    }
}
```

```
double A[9] =
  { 0,   1,   2,
    4,   8,  16},
   32,  64, 128};

double B[3] = A+3;

sum_rows1(A, B, 3);
```

Memory Aliasing (2)

```
/* Sum rows of n X n matrix a
   and store in vector b  */
void sum_rows1(double* a, double* b, long n) {
    long i, j;
    for (i = 0; i < n; i++) {
         b[i] = 0;
         for (j = 0; j < n; j++)
             b[i] += a[i*n + j];
    }
}
```

```
double A[9] =
  { 0,   1,   2,
    4,   8,  16},
   32,  64, 128};

double B[3] = A+3;

sum_rows1(A, B, 3);
```

Value of B:

```
init:  [4, 8, 16]
```

```
i = 0: [3, 8, 16]
```

```
i = 1: [3, 22, 16]
```

```
i = 2: [3, 22, 224]
```

Removing Aliasing

```
/* Sum rows of n X n matrix a
   and store in vector b  */
void sum_rows2(double* a, double* b, long n) {
    long i, j;
    for (i = 0; i < n; i++) {
        double val = 0;
        for (j = 0; j < n; j++)
            val += a[i*n + j];
        b[i] = val;
    }
}
```

```
# sum_rows2 inner loop
.L10:
        addsd   (%rdi), %xmm0       # FP load + add
        addq    $8, %rdi
        cmpq    %rax, %rdi
        jne     .L10
```

Example: Accumulating

```
long add_ints(int* data, int size) {
    int i;
    long total = 0;
    for (i = 0; i < size; i++) {
        total += data[i];
    }
    return total;
}
```

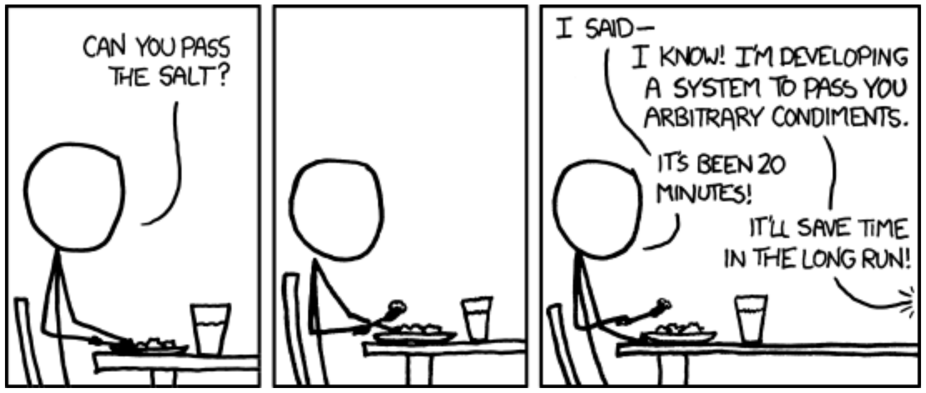
Loop Unrolling

```
long add_ints(int* data, int size) {
    int i;
    long total = 0;
    /* Combine 2 elements at a time */
    for (i = 0; i < size; i+=2) {
        total += data[i] + data[i+1];
    }
    /* Finish any remaining elements */
    for (; i < size; i++) {
        total += data[i];
    }
    return total;
}
```

Loop Unrolling with Separate Accumulators

```
long add_ints(int* data, int size) {
    int i;
    long total1 = 0;
    long total2 = 0;
    /* Combine 2 elements at a time */
    for (i = 0; i < size; i+=2) {
        total1 += data[i];
        total2 += data[i+1];
    }
    /* Finish any remaining elements */
    for (; i < size; i++) {
        total1 += data[i];
    }
    return total1 + total2;
}
```

Premature Optimization


CAN YOU PASS THE SALT?
I SAID—
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!