MITx 6.004.2x
Computation Structures 2: Computer Architecture

Help selfpoised

Course Progress Dates Discussion

Course / 11. Compilers / Lecture Videos (34:57)

Previous ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ Next

LE11.3

Bookmark this page

Calculator

LE11.3.1 Loop optimizations

1.0/1.0 point (ungraded)
In block structured languages such as C or Java, the scope of a variable declared locally within a block extends only over that block, i.e., the value of the local variable cannot be accessed outside the block. Conceptually, storage is allocated for the variable when the block is entered and deallocated when the block is exited. In many cases, this means the compiler is free to use a register to hold the value of the local variable instead of a memory location.

Consider the following C fragment:

```
int sum = 0;
{ int i;
  for (i = 0; i < 10; i = i+1) sum += i;
}
```

If this loop is compiled such that intermediate results are stored in memory, then the resulting compiled code would look like this:

```
     ST(R31,sum)
     ST(R31,i)
_L7:
     LD(sum,R0)
     LD(i,R1)
     ADD(R0,R1,R0)
     ST(R0,sum)
     ADDC(R1,1,R1)
     ST(R1,i)
     CMPLTC(R1,10,R0)
     BT(R0,_L7)
```

If instead the compilation of this code is optimized by using registers to hold the values of the local variables i and sum, then the compiled code would look like this:

```
     MOVE(R31,R2)          // R2 holds sum
     ST(R2,sum)
     MOVE(R31,R1)          // R1 holds i
_L5:
     ADD(R2,R1,R2)
     ADDC(R1,1,R1)
     CMPLTC(R1,10,R0)
     BT(R0,_L5)
     ST(R2,sum)
```

Define a memory access as any access to memory, i.e., instruction fetch, data read (LD), or data write (ST). Compare the total number of memory accesses generated by executing the optimized loop with the total number of memory access for the unoptimized loop.

Unoptimized loop:

Number of instructions in loop: 8 ✓

Number of data accesses per loop iteration: 4 ✓

Optimized loop:

Number of instructions in loop: 4 ✓

Number of data accesses per loop iteration: 0 ✓

Given that the loop is executed 10 times, what is the total number of memory accesses in the unoptimized loop versus the optimized loop?

Calculator

Unoptimized loop: Total memory accesses: 120 ✓

Optimized loop: Total memory accesses: 40 ✓

Some optimizing compilers "unroll" small loops to amortize the overhead of each loop iteration over more instructions in the body of the loop. For example, one unrolling of the loop above would be equivalent to rewriting the program as:

```
int sum = 0;
{ int i;
  for (i = 0; i < 10; i = i+2) { sum += i; sum += i+1; }
}
```

The hand-compiled unrolled loop looks like this:

```
     MOVE(R31,R2)          // R2 holds sum
     ST(R2,sum)
     MOVE(R31,R1)          // R1 holds i
_L5:
     ADD(R2,R1,R2)
     ADDC(R1,1,R0)
     ADD(R2,R0,R2)
     ADDC(R1,2,R1)
     CMPLTC(R1,10,R0)
     BT(R0,_L5)
     ST(R2,sum)
```

Unrolled loop:

Number of instructions in loop: 6 ✓

Number of data accesses per loop iteration: 0 ✓

Unrolled loop: Total memory accesses: 30 ✓

Submit

Discussion

Hide Discussion

Topic: 11. Compilers / LE11.3

Add a Post

Show all posts by recent activity

Memory Access defined
I may have missed its introduction in the lectures, but here "memory access" appears to mean any instruction that accesses a regist... 6

Previous Next

© All Calculator

edX

About
Affiliates
edX for Business
Open edX
Careers
News

Legal

Terms of Service & Honor Code
Privacy Policy
Accessibility Policy
Trademark Policy
Sitemap

Connect

Blog
Contact Us
Help Center
Media Kit
Donate

GET IT ON Google play
Download on the App Store

Calculator