Class exercise: Function inlining and specialization

COMP 40
November 21, 2011

Group

Keeper of the record:

Other group members:

Function inlining

In this problem I want you to estimate the cost of function calls by counting calls, returns, arithmetic operations, loads, compares, and branches.

What’s to be gained by inlining \texttt{UArray\_at(segment, r4)}? Assume that after inlining, the compiler improves the code as much as possible.

```c
void *UArray\_at(T array, int i) {
    assert(array);
    assert(i >= 0 && i < array->length);
    return array->elems + i*array->size;
}
```

Without inlining

<table>
<thead>
<tr>
<th>Calls &amp; returns</th>
<th>Arithmetic</th>
<th>Loads and stores</th>
<th>Comparisons</th>
<th>Branches</th>
</tr>
</thead>
</table>

After inlining and specialization

What’s to be gained by inlining \texttt{Bitpack\_getu(instr, 3, 6)}? Assume that after inlining, the compiler improves the code as much as possible.

```c
static inline uint64\_t shl(uint64\_t word, unsigned bits) {
    assert(bits <= 64);
    if (bits == 64)
        return 0;
    else
        return word << bits;
}

static inline uint64\_t shr(uint64\_t word, unsigned bits) { // shift R logical
    assert(bits <= 64);
    if (bits == 64)
        return 0;
    else
        return word >> bits;
}

uint64\_t Bitpack\_getu(uint64\_t word, unsigned width, unsigned lsb) {
    unsigned hi = lsb + width; // one beyond the most significant bit
    assert(hi <= 64);
    return shr(shl(word, 64 - hi), 64 - width); // different type of right shift
}
```

Without inlining

<table>
<thead>
<tr>
<th>Calls &amp; returns</th>
<th>Arithmetic</th>
<th>Loads and stores</th>
<th>Comparisons</th>
<th>Branches</th>
</tr>
</thead>
</table>

After inlining and specialization

<table>
<thead>
<tr>
<th>Calls &amp; returns</th>
<th>Arithmetic</th>
<th>Loads and stores</th>
<th>Comparisons</th>
<th>Branches</th>
</tr>
</thead>
</table>