Efficiency: caching

COSC 208, Introduction to Computer Systems, 2021-10-29

Announcements

- Project 2 Part 2 due tomorrow at 11pm
- Work on Lab 7
- Attend research talks/teaching demos
    - Today 1:45pm and 2:15pm

Outline

- Warm-up
- Clarifications on Project 2 Part 2
- Instances of caching

Warm-up

- Q1: *Change the registers used in the assembly code to minimize the number of required loads/stores. However, remember to adhere to the calling conventions: namely, a function's parameters are stored in registers `w0`, `w1`, ..., `w7`, and a function's return value is stored in `w0`.*

```
000000000000088c <interest_due>:
    88c:    sub sp, sp, #0x20 XXXXX
    890:    str w0, [sp, #12] XXXXX
    894:    str w1, [sp, #8]  XXXXX
    8a0:    mul **w0 -> w1**, w1, w0
    8a4:    str w0, [sp, #20]   XXXXX
    8a8:    mov w0, #0x4b0
    8b0:    ldr w1, [sp, #20]   XXXXX
    8b8:    sdiv    w0, w1, w0
    8c4:    add sp, sp, #0x20   XXXXX
    8c8:    ret

00000000000008cc <make_payment>:
    8cc:    stp x29, x30, [sp, #-48]!
    8d0:    mov x29, sp
    8d4:    str w0, [sp, #28]
    8d8:    str w1, [sp, #24]
    8dc:    str w2, [sp, #20]
    8e0:    ldr w1, [sp, #20]
    8e8:    bl  88c <interest_due>
    8f0:    ldr w1, [sp, #24]
    8f8:    sub **w0 -> w1**, w1, w0
    8fc:    str w0, [sp, #44]   XXXXX
    900:    ldr w1, [sp, #44]   XXXXX
    904:    ldr **w0 -> w2**, [sp, #28]
    908:    cmp w1, **w0 -> w2**
    90c:    b.le    918 <make_payment+0x4c>
    910:    **str   wzr, [sp, #28] -> mov w0, wzr**
    914:    b   928 <make_payment+0x5c>
    918:    ldr w1, [sp, #28]   XXXXX
    91c:    ldr w0, [sp, #44]   XXXXX
    920:    sub w0, **w1 -> w2**, **w0 -> w1**
    924:    str w0, [sp, #28]   XXXXX
    928:    ldr w0, [sp, #28]   XXXXX
    92c:    ldp x29, x30, [sp], #48
    930:    ret
```

- Sometimes load/stores cannot be eliminated
    - Use a CPU cache to reduce the overhead of loads/stores

Clarifications on Project 2 Part 2

- Forms of `ldp`/`stp` — Dive Into Systems Section 9.2 Table 2
    - `ldp D1, D2, [X0]` — load only
        - `D1 = *X0`
        - `D2 = *(X0+8)`
    - `ldp D1, D2, [X0,#0x10]!` — update then load
        - `X0 = X0 + 0x10`
        - `D1 = *X0`
        - `D2 = *(X0+8)`
    - `ldp D1, D2, [X0], #0x10` — load then update
        - `D1 = *X0`
        - `D2 = *(X0+8)`
        - `X0 = X0 + 0x10`
- Simulator treats spaces as a delimeter
    - e.g., `ldp D1, D2, [X0]` is broken into an array of strings: `char instruction[] = { "ldp", "D1", "D2", "[X0]", NULL }`
- Memory references can be further broken down
    - e.g., `[X0,#0x10]!` is broken into an arrya of strings: `char addr[] = { "X0", "#0x10", NULL }`
- Other questions

Instances of caching

- CPU caches
    - *Why do we have caches on the CPU?* --- accessing main memory is ~100x slower than accessing a register
    - Store instrutions and data (stack, heap, etc.) from main memory
    - Three levels --- L1, L2, and L3
    - Range in size from a few KB to a few MB
    - Cache line (i.e., cache entry) is typically larger than a word -- e.g., 128 bytes
        - *Why?* --- spatial locality
    - What happens when we write to memory?
        - Write through cache --- write to the cache and main memory
        - Write back cache --- initially write to the cache; write to main memory when the entry is evicted from the cache
        - *What are the advantages of each approach?* --- write through cache ensures consistency between CPU cores; write back cache only inccurs the overhead of accessing main memory when absolutely necessary
- Web browser caches
    - *Why do web browsers have caches?*
        - Accessing remote network storage is >50x slower than accessing a solid state drive (SSD)
        - Spatial locality --- many aspects of a web page are also used with other pages on the same site: e.g., images, Cascading Style Sheets (CSS), JavaScript (JS)
        - Temporal locality --- users often visit the same web page repeatedly: e.g., Google
        - Internet Service Provider (ISP) may limit amount of data downloaded/uploaded per month
    - Store static content (e.g., images, CSS, JS)
    - Web browser caches are read-only
- Content distribution networks (CDNs)
    - Collection of geographically distributed servers that delivery content (e.g., streaming videos) to users
    - User's computers contact a server that is "nearby"
        - Ideally measured in terms of latency, which is a function of geographic distance, network routes, and network load
        - Analogy: time it takes to drive somewhere is a function of geographic distance, the route you take, and the amount of traffic on the road

- CDN servers fetch and cache content from origin servers
- Popular content (e.g., image from the front page of the NY Times) is more likely to already be cached

Extra practice

- Q6: *Cross-out unnecessary loads and stores from the assembly code.*

```
000000000000071c <multiply>:
    71c:    d10083ff    sub sp, sp, #0x20
    720:    b9000fe0    str w0, [sp, #12]       XXXXX
    724:    b9000be1    str w1, [sp, #8]        XXXXX
    728:    b9400fe1    ldr w0, [sp, #12]       XXXXX
    72c:    b9400be0    ldr w1, [sp, #8]        XXXXX
    730:    1b007c20    mul w0, w1, w0
    734:    b9001fe0    str w0, [sp, #28]       XXXXX
    738:    b9401fe0    ldr w0, [sp, #28]       XXXXX
    73c:    910083ff    add sp, sp, #0x20
    740:    d65f03c0    ret
```

- Q7: *Cross-out unnecessary loads and stores from the assembly code. Remember to adhere to the calling conventions: namely, a function's parameters are stored in registers `w0`, `w1`, ..., `w7`, and a function's return value is stored in `w0`.*

```
0000000000000744 <volume>:
    744:    a9bd7bfd    stp x29, x30, [sp, #-48]!
    748:    910003fd    mov x29, sp
    74c:    b9001fe0    str w0, [sp, #28]
    750:    b9001be1    str w1, [sp, #24]
    754:    b90017e2    str w2, [sp, #20]
    758:    b9401be1    ldr w1, [sp, #24]               XXXXX
    75c:    b9401fe0    ldr w0, [sp, #28]               XXXXX
    760:    97ffffef    bl  71c <multiply>
    764:    b9002fe0    str w0, [sp, #44]               XXXXX
    768:    b94017e1    ldr w1, [sp, #20]
    76c:    b9402fe0    ldr w0, [sp, #44]               XXXXX
    770:    97ffffeb    bl  71c <multiply>
    774:    b9002fe0    str w0, [sp, #44]               XXXXX
    778:    b9402fe0    ldr w0, [sp, #44]               XXXXX
    77c:    a8c37bfd    ldp x29, x30, [sp], #48
    780:    d65f03c0    ret
```