The Alpha AXP, part 7: Memory access, loading unaligned data

devblogs.microsoft.com/oldnewthing/20170815-00

August 15, 2017

Raymond Chen

Last time, we look ed at loading aligned memory. Now we're going to look at unaligned data.

Let's load an unaligned quad. The unaligned quad will span two aligned quads, so we will need to load two quads, extract the pieces, and merge them together.

```
LDQ_U   t1, (t0)    ; load lower container ; t1 = FEDC BA??
LDQ_U   t2, 7(t0)   ; load upper quad      ; t2 = ???? ??HG
EXTQL   t1, t0, t1  ; align lower portion  ; t1 = 00FE DCBA
EXTQH   t2, t0, t2  ; align upper portion  ; t2 = HG00 0000
BIS     t1, t2, t1  ; combine              ; t1 = HGFE DCBA
```

In the case where the value happens to have been aligned by sheer luck, the operation still works as intended. They do a bunch of redundant work (because they are dealing with a misalignment that never happened), but you still get the correct result.

```
LDQ_U   t1, (t0)    ; load lower container ; t1 = HGFE DCBA
LDQ_U   t2, 7(t0)   ; load upper quad      ; t2 = HGFE DCBA
EXTQL   t1, t0, t1  ; align lower portion  ; t1 = HGFE DCBA
EXTQH   t2, t0, t2  ; align upper portion  ; t2 = HGFE DCBA
BIS     t1, t2, t1  ; combine              ; t1 = HGFE DCBA
```

A similar pattern exists for unaligned longs. Longs require an extra step to ensure the result is in canonical form.

```
LDQ_U   t1, (t0)    ; load lower container ; t1 = BA?? ????
LDQ_U   t2, 3(t0)   ; load upper quad      ; t2 = ???? ??DC
EXTLL   t1, t0, t1  ; align lower portion  ; t1 = 0000 00BA
EXTLH   t2, t0, t2  ; align upper portion  ; t2 = 0000 DC00
BIS     t1, t2, t1  ; combine              ; t1 = 0000 DCBA
ADDL    t1, zero, t1; put in canonical form; t1 = ssss DCBA
```

And you can probably guess the pattern for unaligned words:

```
LDQ_U   t1, (t0)    ; load lower container ; t1 = A??? ????
LDQ_U   t2, 1(t0)   ; load upper quad      ; t2 = ???? ???B
EXTWL   t1, t0, t1  ; align lower portion  ; t1 = 0000 000A
EXTWH   t2, t0, t2  ; align upper portion  ; t2 = 0000 00B0
BIS     t1, t2, t1  ; combine              ; t1 = 0000 00BA
```

If you need sign extension for the unaligned word, then you can use the trick we saw last time.

```
LDQ_U   t1, (t0)    ; load lower container     ; t1 = A??? ????
LDQ_U   t2, 1(t0)   ; load upper quad          ; t2 = ???? ???B
LDA     t3, 2(t0)   ; sneaky trick to extract at index 6+7
EXTQL   t1, t3, t1  ; align lower portion high ; t1 = 0A?? ????
EXTQH   t2, t3, t2  ; align upper portion high ; t2 = B000 0000
BIS     t1, t2, t1  ; combine                  ; t1 = BA?? ????
SRA     t1, #48, t1 ; shift right with sign    ; t1 = ssss ssBA
```

Exercise: There's an obvious continuation of this pattern for unaligned bytes, so why doesn't anybody use it?

That's it for loading bytes, words, and unaligned data from memory. Next time, we'll start looking at writing them, which is a lot more complicated.

Bonus chatter: Later versions of the Alpha AXP processor added support for byte reads and writes, as well as aligned word reads and writes. This makes code easier to write, but probably makes the store-to-load forwarder logic much harder.

Raymond Chen

Follow