Retrocomputing

Why do only the low 7-bits of the R register increment?

Asked 4 years, 6 months ago Modified 5 months ago Viewed 3k times

17

2

On the Z80 the `R` register is 8 bits wide, as evidenced by `LD A,R` and `LD R,A` And every M1 cycle, `R` increments by 1. But the uppermost bit does not participate in this increment. `R` will not go from 0x7F to 0x80 or from 0xFF to 0x00. Apparently that behaviour is useful for DRAM refresh.

But how was that implemented?

- The Z80 has a 4-bit ALU which is used for all 8-bit and 16-bit operations, but apparently not this one. That means there must be a separate incrementer on the die?
- I'm sure the Z80 designers were not imagining a programmer wanting to use R as an 8 bit counter, so they could have saved one bit of memory by having `R` only 7 bits wide.

z80 registers

Share Improve this question Follow

edited Mar 30 at 18:41
user3840170
17.2k 4 66 114

asked Mar 8, 2018 at 21:18
OmarL
34.8k 8 119 242

3 Answers

Sorted by: Highest score (default)

20

The `R` register is solely used to put out consecutive increased row addresses for RAM refresh. The fact that it is readable and setable by programs (*1) is rather a quirk than a feature (*2).

`I` and `R` is a register pair right beside PC; a clever trick to save circuitry:

- Firstly, `I` needs to be placed on the upper address bus, like `PCH`, so putting it alongside `PCH` means the same buffers and signal lines can be used when the upper half of the interrupt service address is to be put out.
- Secondly, for a refresh address it doesn't really matter if it's on the upper or lower address bus half, so the 'hole' beside `I` can be easy be used for another register holding the refresh (row) address, and again the same buffers/lines can be used to deliver it.
- And lastly, putting `R` here allows the usage of the the same incrementer/decrementer circuits as used to handle the PC.

From this it's easy to see that

- `R` doesn't have a specialized 7-bit incrementer; it uses the full 16-bit incrementer as for `PC`, but during operation on `(I)R`, carry over is disabled between bit 7 and 8, so making it work as 7-bit incrementer.
- Since the register block is a symmetric structure of 12 register pairs, leaving out one bit somewhere in the middle wouldn't have saved anything.

So, why just 7 Bits?

When Faggin et. al. designed the Z80 in 1974/75, the only available DRAM with multiplexed addresses (and row refresh) was Mostek's 4 KiBit Chip MK4096, released in 1973. It was an instant success - a good reason for Zilog to include support for the new RAM interface design. The MK4096 needed a 6-bit row address for refresh. At that time 16 KiBit DRAM's were talked about, but no-one had one until Mostek started selling its MK4114 in 1976 - after the Z80 was released. So for this time-frame, the decision of utilizing 7 bits was already looking forward.

Anything bigger was hard to imagine at that time. After all, it took 4 more years, until 1980, to make 64 KiBit DRAM available (which now would require 8-bit refresh counters).

3 "but uses the full 16 bit incrementer as for the PC, but only the lower 7 bits are loaded back" -- according to the analysis from Ken Shirriff, it uses all 16 bits, but the carry signal is gated after bit 7 to prevent any changes further up the chain. – Jules Mar 9, 2018 at 0:31

@Jules you're right, somehow I assumed the load is modiied, not the counter. – Raffzahn Mar 9, 2018 at 7:41

1 But from what you've said here, it would be less effort to make an 8-bit R register than a 7-bit one, which needed extra gating. Why did Zilog not just leave it harmlessly at 8-bit instead? The 7-bit counter value is just as available. Seems like a deliberate hindrance rather than any kind of saving. They weren't so daft that they couldn't imagine 8-bit refresh addresses being used some day :-) It's got to be something else. – TonyM Apr 13, 2018 at 9:22

1 @TonyM you might want to read it again. To increment `R` the 16 bit (PC) incrementer is used. Since the upper half is `I` , any spill over would be harmful. Ofc, they could have added some gates to disable carry after bit 8. Just the structure isn't that simple. To keep up with the speed needed, it's build of 6 two bit incrementers and a 7/5/3 carry look ahead . It would require several gates to make it break at bit 8, including a different design for one of the 2 bit incrementers. But, supressing carry after bit 7 is just one more input to a large OR - so not even a gate but only one resistor. – Raffzahn Apr 13, 2018 at 9:39

1 Ah, that's quite a different explanation than your answer: under "So, why just 7 Bits?" you speculate about marketing reasons instead. Personally, I've never bought the idea that they limited it because they thought no-one would need it. The 'short-sighted engineers of the past' as seen from the present are recurring theories for these sorts of things. It was going to be a technical limitation that caused it. And in all my years of reading a fair old amount about the Z80, I don't think I've seen text from a Zilog explaining why. I always assumed it just made the IC layout easier. – TonyM Apr 13, 2018 at 11:07

12

The 'R' register increment happens simultaneously with other operations, as it happens for every instruction, but must occur *after* the refresh cycle in each instruction (which happens at the end of instruction fetch, thus pushing the only time the increment can actually happen into the time when the fetched instruction is executing), so it cannot use the standard ALU. Therefore, the Z80 has a separate 7-bit wide incrementer for the R register. It is implemented as part of the circuitry for the instruction counter incrementer (instruction counter increment can happen immediately after instruction fetch, i.e. while the refresh cycle is being generated, so doesn't conflict with ordinary instruction execution, but the ALU incrementer is only 8 bits I believe, while the instruction counter needs a full 16 bit incrementer).

7 bits would have been chosen because 128 was the number of refresh cycles required for the most common DRAM chip type at the time the Z80 was designed, the 4116.

Ken Shirriff has die photos and a more detailed explanation on his blog, and a detailed article about the implementation of the incrementer. It's also worth noting that other register increments are also apparently implemented not using the ALU but by a dedicated increment/decrement circuit. Presumably, because this doesn't need full carry prediction, but a much simpler circuit to detect the case of all ones (or zeroes for decrement) to the right of each bit, this could be done in a single cycle rather than requiring two to feed the operation through the ALU, without wasting too much die space.

Share Improve this answer Follow

edited Mar 9, 2018 at 0:29

answered Mar 8, 2018 at 21:32

Jules

12.4k 2 38 62

Nice photos. Makes me wonder how hard it would have been for the Z80 to exploit the 16-bit bus for 16-bit LD opcodes in the range ED00-ED3F, with operands chosen from (AF,BC,DE,HL,IX,IY,SP,PC). While a 16-bit "LD HL,BC" instruction might not have had much advantage over the two 8-bit instructions "LD H,B / LD L,C", having 16-bit loads available for IX and IY would have made them much more useful. – supercat Mar 8, 2018 at 22:22

1 I suspect the answer lies in this: "And I remember many meetings when we were working on the instruction set. And Federico would say, 'It really needs this one,' and Shima would say, 'No! It adds two more microns.'" – Jeremy Mar 11, 2018 at 17:43

@Jeremy: Heh. I wonder how much the Z80's size would be affected by some of the instruction set features I've wanted (e.g. a mode which would re-purpose six of the "LD B,B", "LD C,C", etc. opcodes as prefixes which that would modify most instructions that would use "A" to use the indicated register instead, a tweak to LDIR that would avoid the instruction re-fetch unless an interrupt was pending, or add/subtract SP,imm8). – supercat Mar 13, 2018 at 16:52

6

@Jules's link to the incrementer details offers an unsatisfying explanation:

> When the Z-80 was introduced, 16K memory chips were popular. Since they held 2^14 bits, they had 7 row address bits and 7 column address bits. Thus, a 7 bit refresh value matched their need. Unfortunately, this rapidly became obsolete with the introduction of 64K memory chips that required 8 refresh bits.

That explanation suggests no reason why the designers couldn't have chosen 8 bits, with an eye to future memory devices.

I think the real reason lies elsewhere in that description.

Firstly:

> Bits 7 through 11 are computed using the carry lookahead value, allowing them to be computed without waiting on the low-order bits. In parallel, the carry/borrow out of these bits is computed by the large NOR gate in the middle, and used to compute bits 12 through 14. The last carry lookahead value is computed at the left and used to compute bit 15. Note that the number of carry blocks decreases as the number of carry lookahead gates increases. For example, output 6 depends on three inc/dec blocks and no carry lookahead gates, while output 14 depends on one inc/dec block and two carry lookahead gates. If the inc/dec blocks and carry lookahead gates require approximately the same time, then the output bits will be ready at approximately the same time.

i.e. The splits in the circuit structure were designed to meet overall timing constraints on the 16-bit incrementer, without regard for any secondary function it may be used for. As it turned out, one of those splits was between bits 6 and 7.

Next:

> One unexpected feature of the Z-80's incrementer is that it can pass the value through unchanged. If the carry-in to the incrementer/decrementer is set to 0, no action will take place. This seems pointless, but it actually useful since it allows a 16-bit value to be latched and then read back unchanged.