Forum Search:

Forum Search

RetroBrew Computers Forum
Discussion forum for the RetroBrew Computers community.

- Members
- Search
- Help
- Login
- Home

Home » RBC Forums » General Discussion » **Z80 programming techniques**

Show: Today's Messages :: Show Polls :: Message Navigator

TREE VIEW NEW TOPIC POST REPLY

Z80 programming techniques [message #4077] Thu, 11 January 2018 07:32

plasmo

Messages: 604 Senior Member
Registered: March 2017
Location: New Mexico, USA

There are so much Z80 knowledge in here and vcfed that I really want to know more about it and learn from the experts. I know nothing of Z80. I can handle the hardware but I need guidance with software. My end target is Z280, but I'll start with Z80. I'd appreciate recommendations on books/articles about Z80 assembly programming techniques.
Bill

[Updated on: Thu, 11 January 2018 07:34]

Report message to a moderator

PROFILE PM REPLY QUOTE

Re: Z80 programming techniques [message #4078 is a reply to message #4077] Thu, 11 January 2018 11:03

jdgabbard

Messages: 66 Member
Registered: March 2016

Probably the first most important thIng to understand is structure. You have to learn good programming structure. Languages such as BASIC can help, but it's crucial to understand this. After than you have to understand that with assembly, you have to give it structure, as it has none of its own. After than you just have to learn the instructions and what they do at the metal.

Doug Gabbard
Creator of the G80-S Micro Computer and 'Porter' of TinyBASIC 2.5g
Website: http://retrodepot.net
z80 TinyBASIC 2.5g: http://retrodepot.net/?p=424
AtariAge Username: jdgabbard

Report message to a moderator

profile pm reply quote

Re: Z80 programming techniques [message #4080 is a reply to message #4077] Thu, 11 January 2018 11:40

wsm

Messages: 173
Registered: February 2017 Senior Member
Location: AB, Canada

I have two "go to" references for the Z80: the "Z80 Family CPU User Manual" by Zilog and "Programming the Z80" by Rodney Zaks. The former is available for download from the Zilog site. My main and extremely handy reference is the Mostek Z80 Microcomputer System "Micro-Reference Manual" which is pocket sized and has all the instructions, flag changes etc. Unfortunately, I highly doubt whether copies are available.

With any new CPU / architecture, I find it easiest to first look at the register set and understand how they can be manipulated. Load / store to memory, register to register, indexing etc. After that, I then go through the full instruction set to look for unique instructions such as the very useful LDIR and DJNZ instructions, bit manipulation, I/O etc.

There are some nice Z80 extensions from the 8080 for addressing and indexing but the programmer has to make a conscious decision whether they're writing generalized code or Z80 unique code. I tend to always write Z80 code so I also use the Zilog mnemonics versus the Intel mnemonics.

With the Z80 there are a couple of potential "gotchas" when you're first programming it. All of the load / store instructions use the LD mnemonic which can be misleading at first i.e. LD register from memory and LD memory from register. Note that memory locations are usually enclosed in brackets versus unbracketed for constants. Another area to watch out for is whether instructions modify the flag register bits or not.

The Z80 has some interesting instructions such as swapping data with the top of the stack "EX (SP),HL" etc. but I would start by first using the more basic instructions and then possibly going back over the code at a later time to see if it can be optimized. As you get more familiar with the Z80 there becomes a question of whether to actually use the alternate register set in application code. Since I tend to write interrupt driven code, I usually reserve the alternate register set for faster interrupt entry / exit. Rather than a bunch of PUSH / POPs, simply an "EX AF,AF'" and "EXX" gives the interrupt routine a full set of unique registers.

Report message to a moderator

profile pm reply quote

Re: Z80 programming techniques [message #4081 is a reply to message #4080] Fri, 12 January 2018 02:58

tor

Messages: 25
Registered: April 2016 Junior Member
Location: Norway/Japan

I saw the Micro-Reference Manual in scanned PDF form over on classiccmp.org, in the trs80 part of the cpm archive.
And Zaks has given permission to Gaby to distribute his 'Programming the Z80':
http://gaby.de/z80/zaks.html
(I bought the hardcopy back in the eigthies - I've always liked the Zaks books. They're a pleasure to read.)

[Updated on: Fri, 12 January 2018 03:01]

Report message to a moderator

profile pm reply quote

Re: Z80 programming techniques [message #4084 is a reply to message #4081] Fri, 12 January 2018 08:21

plasmo
Messages: 604
Registered: March 2017
Senior Member
Location: New Mexico, USA

Thank you all for the feedback. I downloaded both Z80 Family CPU User Manual as well as Programming the Z80. I also found the Mostek micro-reference manual in archive.org.

LDIR and DJNZ are cool, remind me of decrement and branch (DBcc) of 68000. I don't know how the alternate registers are used in Z80 programming convention, but context swapping seems reasonable--unless the interrupts are nested.

OK, this should be enough to get me started.

Report message to a moderator

profile pm reply quote

Re: Z80 programming techniques [message #4085 is a reply to message #4084] Fri, 12 January 2018 10:08

wsm
Messages: 173
Registered: February 2017
Senior Member
Location: AB, Canada

Quote:

> I don't know how the alternate registers are used in Z80 programming convention, but context swapping seems reasonable--unless the interrupts are nested.

Absolutely correct, but that is where conventions come in. Due to multi-tasking and real-time systems, a concept I learned MANY years ago for interrupts was first level handlers and second level handlers. First level interrupt handlers are not nested and simply do the basic acknowledge and very simple processing in order to remain disabled for minimal elapsed time whereas second level handlers have interrupts enabled and do more extensive data processing. I tend to let the first level handlers just set a flag that the second level handlers (main routines or scheduler) can wait on. The two handlers can also use FIFO queues between them in order to minimize delays and maximize throughput.

Unfortunately there are no universal conventions for the Z80 alternate register set so Z80 interrupt routines are one of the few places I use simple macros for the entry / exit sequence. For example, to switch my Z80 code from a basic 8080 environment (single register set such as CP/M) to a Z80 environment where applications use both register sets just requires the macros to be updated and a re-assembly rather than trying to update all the various interrupt routines which change the register context.

Enjoy the new architecture and have fun. If you're thinking of a new system, I would seriously look at the Z180 with it's integrated peripherals, DMA and some extra instructions (including multiply) versus the basic Z80.

Since I have an original Mostek reference I wasn't aware of how available they are ... they're even on Ebay, albeit very expensive.

Report message to a moderator

profile pm reply quote

Re: Z80 programming techniques [message #4086 is a reply to message #4085] Fri, 12 January 2018 13:06

plasmo

Messages: 604 Senior Member
Registered: March 2017
Location: New Mexico, USA

That makes sense. The high priority interrupt should perform time critical responses and quickly release the CPU for lower priority tasks. So masking off all interrupts and using alternate register set should only be allowed for the high priority interrupts.

My intention is to stay in the Z80 environment so I won't confuse myself with 8080 mnemonic. I wasn't planning on Z80. I want to do Z280. It is a 16-bit machine with interesting capabilities. I believe a very small and cheap Z280 (perhaps I should call it TinyZ) SBC is possible.

Report message to a moderator

profile pm reply quote

Re: Z80 programming techniques [message #4111 is a reply to message #4086] Wed, 17 January 2018 03:53

positron

Messages: 78 Member
Registered: November 2015
Location: Buenos Aires, Argentina

Hi Bill, the attached Zilog document on Interrupt handling could be useful.

Cheers, positron.

- Attachment: The Z80 Family Program Interrupt Structure (Zilog 1978).pdf (Size: 1.20MB, Downloaded 161 times)

Report message to a moderator

profile pm reply quote

Re: Z80 programming techniques [message #4114 is a reply to message #4111] Wed, 17 January 2018 12:36

etchedpixels

Messages: 259 Senior Member
Registered: October 2015

I dread to think how long I've been doing Z80 - too long. Z280 is very different though - you've got semi-sane pointer/offset and stack referencing For Z80 the number one rule is 'keep it in a register'. The number two rule is 'if it won't fit in a register put it at a fixed address'

LDIR is nice but an unrolled loop of LDI statements is faster (and abusing push/pop faster yet!

The original idea of the alternate registers was apparently fast interrupt handling and some systems (eg the

ZX81) use this. In general though I find they are more useful for 32bit maths because you can do things like

```
add hl,de
exx
adc hl,de
exx
```

and for getting enough registers to do things like proportional font rendering.

You don't actually need to write a lot of Z80, for most things not using 32bit maths the SDCC compiler output is sufficiently good.

There is a lot of good Z80 material on the internet, from fast square root routines to floating point maths.

[Updated on: Wed, 17 January 2018 12:43]

Report message to a moderator

profile pm reply quote

Re: Z80 programming techniques [message #4117 is a reply to message #4114] Wed, 17 January 2018 17:09

plasmo

Messages: 604
Registered: March 2017 Senior Member
Location: New Mexico, USA

I come to the party 30 years late to find Z280 quite an interesting processor. The integration of peripherals are expected, but cache, instruction pipelining, burst fill, MMU and traps are quite modern, at least on paper. Booting off serial port is a neat feature I'm going to use. I also want to play a bit with the multiprocessing features. I purchased 5 chips from utsource quite cheaply ($2.14 each) and will have pc boards coming in next week to check them out. This will be quite a departure from 680x0.

Report message to a moderator

profile pm reply quote

Re: Z80 programming techniques [message #4123 is a reply to message #4114] Thu, 18 January 2018 09:11

wsm

Messages: 173
Registered: February 2017 Senior Member
Location: AB, Canada

Quote:

> LDIR is nice but an unrolled loop of LDI statements is faster

The classic time versus space problem, especially when doing something like initializing a large block of memory.

One of the reasons I like the Z180 is it's integrated peripherals including a memory to memory DMA controller. Even with DMA register setup, the break-even compared to LDIR is about 20 bytes. For large areas, DMA only takes 43% as long as LDIR and is twice as fast as multiple LDI's. On my 33MHz Z180 system I developed a CPLD based DMA controller that is about 9 times faster than the Z180's DMA or

about 21 times faster than LDIR. I haven't worked out the relative increase for the Z280 DMA controllers versus LDIR.

Do I still use LDIR? Lots of times I find there are variable length moves and I'll also openly admit that sometimes when I'm programming I get lazy and it's easier than writing the code to calculate 20-bit addresses for DMA. Likewise, IX & IY indexing may be "expensive" but they can be very handy on the Z*80 with it's limited number of registers.

I also dread to think about how long I've played with the Z80 ... let's just say that I started when the first S100 cards came out with them. I tend to write a lot of code in assembler and there's been excursions from the Z-world for several different architectures but I always seem to come back to it. As to assembler versus higher level ... I think it all depends on the kind of code you're writing. I tend to use assembler since I'm usually writing low level code like I/O drivers and BIOS's that are performance oriented and targeted for only one architecture. For code that requires portability, heavy math or graphics it's much easier to use a higher level language.

I'm not trying to belabour this one instruction. I think it just points out that every application is unique and programmers adopt their own styles and conventions which can vary depending upon the requirements.

Report message to a moderator

profile pm reply quote

Re: Z80 programming techniques [message #4124 is a reply to message #4117] Thu, 18 January 2018 10:12

wsm

Messages: 173
Registered: February 2017 Senior Member
Location: AB, Canada

Quote:

> I come to the party 30 years late to find Z280 quite an interesting processor. The integration of peripherals are expected ...

Integrated peripherals definitely reduce the chip count and create standardized I/O routines. One project that I considered but never built was a 5 chip Z180 system: 33MHz Processor, 512KB SRAM, 512KB flash, 74LVC1G19 decoder & RS-232 or USB translator. That plus a few discrete components would create a fast CP/M machine with RAM & flash "disks" plus two serial ports.

The eZ80 is another device that is on my "some day" list. It has highly integrated peripherals and is the equivalent of a Z80 at about 200 MHz ... like a Z280 on steroids. The hardware wouldn't be too difficult but I know there would be a LOT of software development for my purposes.

Not necessarily "retro" hardware projects but using CP/M or MP/M might be considered so. I tend to "play" with older architectures but pushing their performance boundaries within the specifications.

Report message to a moderator

profile pm reply quote

Re: Z80 programming techniques [message #4125 is a reply to message #4124] Thu, 18 January 2018 20:18

plasmo Senior Member

Messages: 604
Registered: March 2017
Location: New Mexico, USA

I love the fact there are so much knowledge & supports in the Z80 community. Every Z80 idea probably has been explored multiple times, yet newbie like me want to try it again, anyway.

The Z280 computer I want to try is a 'disposable' design that is cheap, easy to build and with many programmable I/O so I can experiment with it and then leave it behind afterward. No need to tear down the experiments to salvage the computer. It is just a disposable learning platform.

The design I'm thinking of is a Z280, CPLD (my favorite EPM7128), 1megx16 DRAM (HY5118164), 24MHz clock, real time clock (optional), and compact flash. No ROM because of the cost and hassles of programming ROM. It will boot off CF, so different CF disks boot to different configurations. With the 100-pin CPLD there should be enough spare pins left over to do whatever I like.

Report message to a moderator

profile pm reply quote
tree view new topic post reply

Previous Topic: European distribution of Retrobrew boards
Next Topic: Minimal 8088 SBC Project in Mind

-=] Back to Top [=-
[Syndicate this forum (XML)] [ ] [ ]

Current Time: Mon Jan 25 14:51:07 PST 2021

Total time taken to generate the page: 0.01795 seconds

.:: Contact :: Home ::.