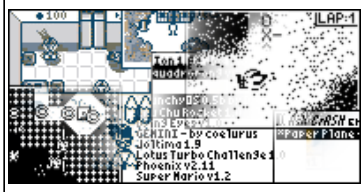

MaxCoderz
for your 1 bit pleasure!

Crazy Z80 optimization trick!

Moderator: MaxCoderz Staff

NewTopic PostReply **Page 1 of 1** [14 posts]

Print view **Previous topic | Next topic**

Author	Message
Dwedit	**Post subject:** Crazy Z80 optimization trick! **Posted:** Sat 24 Nov, 2007 7:08 pm

Offline

Maxcoderz Staff

Joined: Wed 15 Dec, 2004 6:06 am
Posts: 579
Location: Chicago!

Bregalad on the Nesdev forums just informed me of a trick for optimizing if-then-else type blocks, where the "else" area consists of a 2-byte instruction.

So you normally have an if-else-endif block like this:

Code:

```
jr nz,else    ;the IF
;some code
jr endif
else:
;some code
endif:
```

But here's a crazy trick for when the Else code is a single 2-byte instruction:
You use the first byte of a 3 byte instruction with no side effects instead of the "jr endif" line!
So if you had code like this:

Code:

```
cp 7
jr nz,else
ld a,3
jr endif
else:
ld a,4
endif:
```

You could replace it with this:

Code:

```
cp 7
jr nz,else
ld a,3
.db $C2  ;jp nz,xxxx
else:
```

```
ld a,4
endif:
```

Instead of branching over the ld a,4 instruction, it now executes a jp nz,XXXX instruction where the XXXX is the two bytes of the next instruction. You already know what the flags will be here, so you can make the jump never taken. You can use this to skip the next two bytes of execution! Who needs to branch over it?

You know your hexadecimal output routine is broken when it displays the character 'G'.

Top | Profile | Quote

King Harold

Offline

Calc King

Joined: Sat 05 Aug, 2006 7:22 am
Posts: 1513

Post subject: **Posted:** Sat 24 Nov, 2007 8:03 pm

omg that is cool!
what would that do to a disassembler?

Top | Profile | Quote

CoBB

Offline

MCF Legend

Joined: Mon 20 Dec, 2004 8:45 am
Posts: 1601
Location: Budapest, Absurdistan

Post subject: **Posted:** Sat 24 Nov, 2007 9:50 pm

Nice idea. This could also be done for a one-byte else block using jr. And theoretically for a 3-byte block too (as long as the side effects are acceptable), but that could in no way be faster than branching directly.

The Independent Z80 Assembly Guide
Acelgoyobis
PindurTI

Top | Profile | Quote

Liazon

Offline

Calc Guru

Joined: Thu 27 Oct, 2005 8:28 pm
Posts: 962

Post subject: **Posted:** Sun 25 Nov, 2007 3:46 am

o.O wow i'm speechless...

Image Image Image

Top | Profile | Quote

blueskies

Offline

Calc Wizard

User avatar

Joined: Tue 25 Apr, 2006 2:24 pm
Posts: 553

Post subject: **Posted:** Sun 25 Nov, 2007 4:07 am

what, you guys didn't know about this?

j/k, I don't even understand.

Top | Profile | Quote

King Harold

Post subject: **Posted:** Sun 25 Nov, 2007 11:09 am

the instruction you branch to is the address - part of the other jump, which should not be taken (because

Offline

Calc King

Joined: Sat 05 Aug, 2006 7:22 am
Posts: 1513

the condition is never true - so the instructions in the first part should not have an unpredictable result) so those 2 bytes are skipped without a jump (they are loaded as address that is never used)

right?

Top Profile Quote

driesguldolf **Post subject:** **Posted:** Sun 25 Nov, 2007 1:56 pm

Offline

Extreme Poster

Joined: Thu 17 May, 2007 4:49 pm
Posts: 395
Location: $4080

That is one cool trick!

@King Harold: that wouldn't harm a disassembler at all, you just won't be able to see the else-block.

Top Profile Quote

King Harold **Post subject:** **Posted:** Sun 25 Nov, 2007 2:50 pm

Offline

Calc King

Joined: Sat 05 Aug, 2006 7:22 am
Posts: 1513

Unless it takes the first jump and reads those instruction and them reads the instructions without taking the jump and then having a double instruction on some addresses? (would that happen?)

Top Profile Quote

Dwedit **Post subject:** **Posted:** Sun 25 Nov, 2007 3:02 pm

Offline

Maxcoderz Staff

Joined: Wed 15 Dec, 2004 6:06 am
Posts: 579
Location: Chicago!

I think the disassembler I made would interpret it as a 3 byte instruction, and set the else label to be relative to an instruction boundary.

You know your hexadecimal output routine is broken when it displays the character 'G'.

Top Profile Quote

driesguldolf **Post subject:** **Posted:** Sun 25 Nov, 2007 6:01 pm

Offline

Extreme Poster

Joined: Thu 17 May, 2007 4:49 pm
Posts: 395
Location: $4080

Image
PTI is always correct

j/k, I guess it's disassembler specific

Edit:
Image
I guess some emulators just don't have it...

Top Profile Quote

CoBB **Post subject:** **Posted:** Sun 25 Nov, 2007 6:58 pm

Offline

MCF Legend

Joined: Mon 20 Dec, 2004 8:45 am
Posts: 1601
Location: Budapest, Absurdistan

> **driesguldolf wrote:**
>
> PTI is always correct

But that's only possible because the runtime value of PC is available to the emulator, while an offline disassembler won't be able to analyse the code at such depth. I added that feature to make disassembly more robust (e.g. legitimate instructions can be masqueraded similarly if there are some data bytes before them). The fact that it works for this trick is just a direct consequence of that.

The Independent Z80 Assembly Guide
Acelgoyobis
PindurTI

Top | Profile | Quote

qarnos

Post subject: Re: Crazy Z80 optimization trick!

Posted: Tue 27 Nov, 2007 7:56 pm

Offline

Maxcoderz Staff

Joined: Thu 01 Dec, 2005 9:04 am
Posts: 227
Location: Melbourne, Australia

> **Dwedit wrote:**
>
> Instead of branching over the ld a,4 instruction, it now executes a jp nz,XXXX instruction where the XXXX is the two bytes of the next instruction. You already know what the flags will be here, so you can make the jump never taken. You can use this to skip the next two bytes of execution! Who needs to branch over it?

Cool idea, but from all sources I can find (here's one) the **JP cc** instructions take 10 T-states regardless of whether or not the jump is actually taken, so this trick would be no different, timing wise, than changing **JR endif** (which takes 12 T-states) to **JP endif**.

It does, however, save you one byte and 2 clocks over **JR endif**, and two bytes over **JP endif** but for the sake of code readability I probably wouldn't bother!

"I don't know why a refrigerator is now involved, but put that aside for now". - *Jim e* on unitedti.org

avatar courtesy of driesguldolf.

Top | Profile | Quote

tr1p1ea

Post subject:

Posted: Wed 28 Nov, 2007 6:57 am

Offline

Maxcoderz Staff

Joined: Thu 16 Dec, 2004 10:06 pm
Posts: 4108
Location: I cant seem to get out of this cryogenic chamber!

Pretty clever trick, would probably only use it in size critical routines however.

"My world is Black & White. But if I blink fast enough, I see it in Grayscale."

Top | Profile | Quote

qarnos

Post subject:

Posted: Fri 30 Nov, 2007 8:49 am

Offline

Maxcoderz Staff

Joined: Thu 01 Dec, 2005 9:04 am
Posts: 227
Location: Melbourne, Australia

Now that I think about it, this idea *does* offer a time benefit if you are talking about a 1 byte instruction, instead of two.

The **JR** instruction takes only 7 T-States if the branch isn't taken (presumably because the Z80 doesn't have to add the relative offset to PC).

Compare this code:

Code:

```
        jp  z, _else    ; [10]
        add hl, bc      ; [11]
        jp  _endif      ; [10]
_else:  add hl, de      ; [11]
_endif:
```

That takes 31 T-states for *if* and 21 T-states for *else*.

Now try this:

Code:

```
        jp  z, _else    ; [10]
        add hl, bc      ; [11] assume this can't ever carry
        .db $38         ; [7] code for JR, C
_else:  add hl, de      ; [11]
```

This only takes 28 T-states for *if*. A small saving, but could be useful in tight loops, and saves 2 bytes!

The only reason not to use this for 1-byte instructions would be code readability and bug safety. Watch those flags!

_____

"I don't know why a refrigerator is now involved, but put that aside for now". - *Jim e* on unitedti.org

avatar courtesy of driesguldolf.

Top Profile Quote

Display posts from previous: All posts Sort by Post time Ascending Go

NewTopic PostReply **Page 1 of 1** [14 posts]

Board index » Programming » Programming Help All times are UTC

Who is online

Users browsing this forum: No registered users and 2 guests

You **cannot** post new topics in this forum
You **cannot** reply to topics in this forum
You **cannot** edit your posts in this forum
You **cannot** delete your posts in this forum
You **cannot** post attachments in this forum

Search for: Go Jump to: Programming Help Go