

Re: Painless micro program

5:54
pm,
krw
<k...@xxxxxxxxxxx>
wrote:

In
article
<1179616199.332790.162...@xxxxxxxxxxxxxxxx
kensm...@xxxxxxxxxxx
says...

[...
CDP1802
...]

Well,
the
8051,
at
least
in
its
original
incantation,
isn't
much
of
a
screamer
either
(12
clocks
per
op,
as
well).

There's
a
big
difference
between
12
cycles
of
a
12MHz
clock

Re: Painless micro program

and
12
cycles
of
a
2
MHZ
clock.
Besides:

Like
I
said,
there
isn't
much
of
a
difference.

A
=
B
+
C

Compiles
to
something
like
16
instructions
if
A,
B
and
C
are
randomly
located
in
the
RAM
area.

Re: Painless micro program

The
8051
is
not
so
different
if
A,
B,
and
C
are
randomly
located
in
external
RAM.
I'd
never
use
a
HLL
on
such
a
beast.

```
MOV
DPTR,#B ;
1
MOVX
A,@DPTR ;
2
MOV R5,A
; 3
MOV
DPTR,#C ;
4
MOVX
A,@DPTR ;
5
ADD A,R5
; 6
MOV
DPTR,#A ;
7
MOVX
@DPTR,A ;
```

Re: Painless micro program

8

The 8051
took 8. Now
lets see if I
can
remember
the 1802
well
enough:

LD
#LOW(B) ;
1 Data =
low 8 of
address
PLO R5 ; 2
Put to low 8
LD
#HIGH(B) ;
3 Data =
high 8 of
address
PHI R5 ; 4
Put to high
8
LD
#LOW(C) ;
5
PLO R6 ; 6
LD
#HIGH(C) ;
7
PHI R6 ; 8
LD
#LOW(A) ;
9
PLO R7 ;
10
LD
#HIGH(A) ;
11
PHI R7 ; 12
LDN R5 ;
13 Load
what R5
points to
SEX R6 ;
14 Do math
with what
R6 points to

Re: Painless micro program

5

Re: Painless micro program

ADD ; 15
Do the math
STD R7 ;
16 Store
where R7
points

The
instruction
count for
the 1802 is
double that
of the 8051
and with
the slower
clock speed,
the 1802
takes about
a billion
times longer
to
do the same
operation.

Impressive! For a
6800-family gadget, it
would be

LDAA A
ADDA B
STAA C

And on the 6803, using
"LDAD..." (load double etc)
would do it for
word operands, using the A
and B accumulators as a
16-bit register.

On a PDP-11, it would have
been...

MOV.B A, C or just MOV
for word data
ADD.B B, C

and the MSP430 is
essentially the same,
executing the sequence in

Re: Painless micro program

about 400 ns. Both machines allow a choice of relative (relocatable) or absolute addressing. The rumor is that a few guys invented the PDP-11 architecture one night in Gordon Bell's basement.

On a 68K, it would be three ops, loadreg/add/store, but you'd have your choice of byte, word, or long.

There weren't many triple-operand machines around, ever.

John

I actually have worked with a machine that used up to 7 operands for a single instruction. IIRC the instruction was a repeated move selective replace using different source and destination indexes, different base address (with a special interpretation for one), a mask value, replace value, and iteration count.

Yikes!

Some of the CISC-ier machines, like HP 3000 and Vax and the nearly forgotten, failed 32-bit Intel chip (what was that called?) had outrageous instructions, like polynomial eval and hairy string ops, but that was just tons of microcode so doesn't really count, by my official decree. The 68332 even has a few, like the linear interpolation thing and hi/lo limit checks.

John

I think the old intel processor you are talking about the i432 "MicroFlame".

Re: Painless micro program

iAPX432.

IIRC it was a four or five chip group for the processor (kinda like a 2901 bit slice machine) and about a 11 by 14 board full of TTL logic (kinda like the old IBM PC or XT).

Three (monstrous) chips for the CPU, IIRC.

Interesting that the chip was sunk by its architectural philosophy. It was designed near the peak of the Pascal craze, wasn't it? Similar to the Western Digital p-machine and the HP3000. HP over-reacted and next did the very risc-y HPPA, which originally didn't even have a multiply, in the name of intellectual purity.

There seems to be an erratic relationship between instruction set/memory architectures and market success. Some truly horrible architectures have been big hits (IBM character-oriented machines, PDP-8, x86, 8048/51, PIC) and some elegant ones have been hits, too (S/360, PDP-11, Sparc, PPC). Itanic is still up for grabs, which is interesting because of how many presumably smart people have invested how many billions of dollars so far.

John

.