Timing in Sigma

Feb 27, 1959

Talk with Ed Voorhees, Erich Bloch, Bob Blosk, Bill Madden

(Bloch)

I-Box 0.4 μs } Sigma } all in synchonism
Lookahead 0.2 μs
Mem bus 0.2 μs

VFL Cycle 0.45 μs not in synchonism
Time for 1 byte

F. Point no cycle as such -

average case: no more than 4 bit normalize

Add 0.95
Mpy 1.75
Div. 7.0

[Load 0.45
mantissa add 0.45
Store 0.45

Execution of instruction.
Fetch from LA or store to LA

checking of instr is overlapped with next instr (0.2 μs clock)

no addressable regs. are modified.

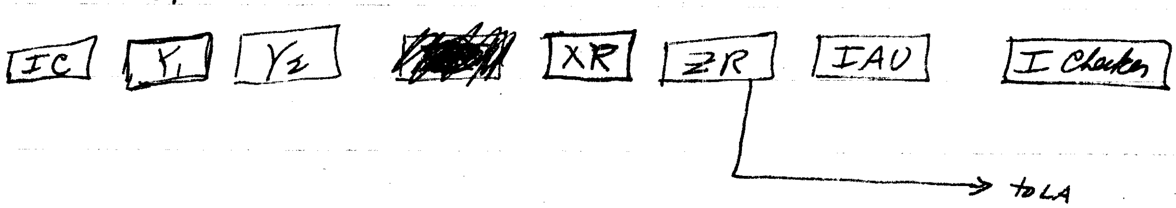
Add	Mpy
	prep of mcnd
	checking of add

{ Exponent differences in add

(Blosk) I Box :

parity prediction check
1/2 wd adder

64 bits

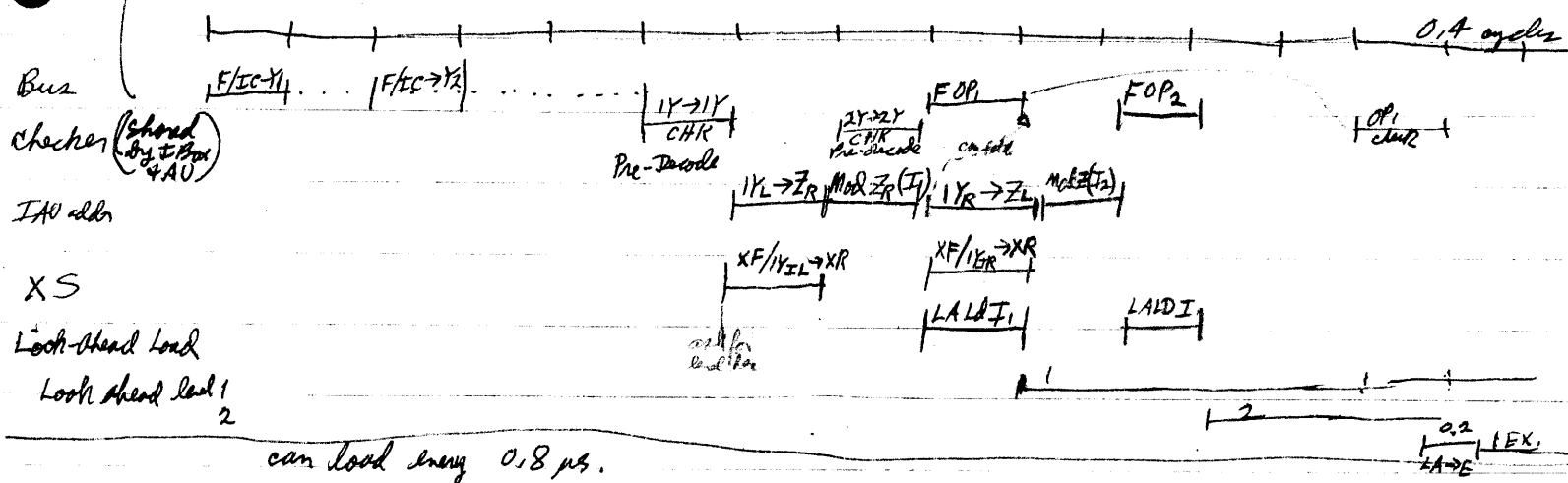


F/IC-1Y

AU has another checker

1.6

if error - repeat + do check cycle



can load every 0,8 μs.

1. going into ZL gives signal that emptying a Y reg
2. operand fetch takes priority after instr. fetch.

IC controls
mod controls
LA load controls

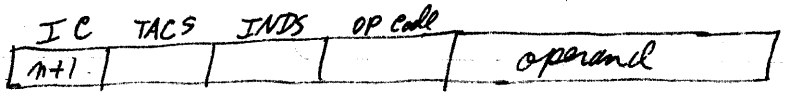
3. Data from X mem takes one cycle longer on 1.

4. on checker: LA gets priority over I Box.

½ μs	2 μs Mem	
0.4	0.4 μs	IBox → BCU
0.2	0.2	BCU
0.6	1.0 μs	Mem readout 1.0 to 1.2
0.2	0.2	Mem → IBox
1.4	1.8 – 2.0	still 2.0 μs

IBox 1.6 μs 0,4 μs faster. - on start up + branches.
but may be held up by conflicts.

in LA:



INDS can be set by fetches, E box, etc.
when goes into Arith Unit - can cause interrupt.

on an interrupt:

1. reset controls - stops what is being done
2. look for pseudo stores - if any do "house cleaning" replace old values.
3. fetch new instr

On instr. fetch: non-correctible (more than 1 bit) errors & correctible errors:

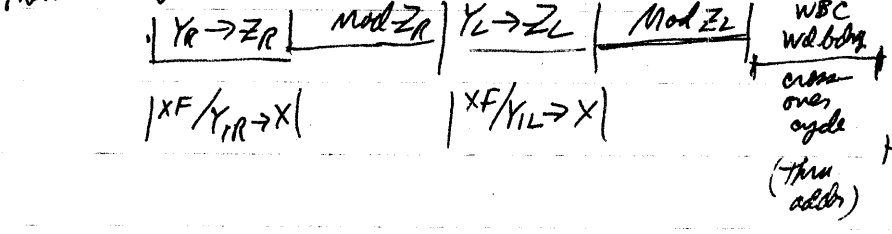
if correctible: Take correct-cycle (0.4 μs more)

if not correctible: Take correct-cycle - Then find multiple error - turn on tag bit on data wd. - goes on thru - is carried to LA & causes regular interrupt.

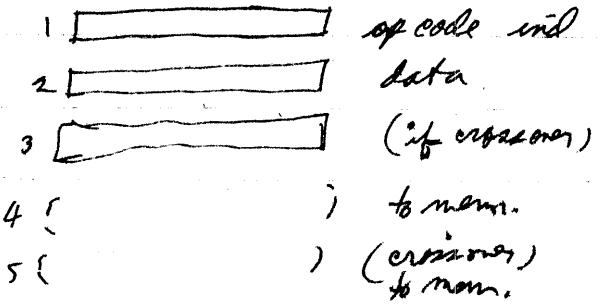
Branch - wrong way. - same as interrupt in timing.

Full wds go in $L \to Z_L$, $R \to Z_R$

Normal indexing

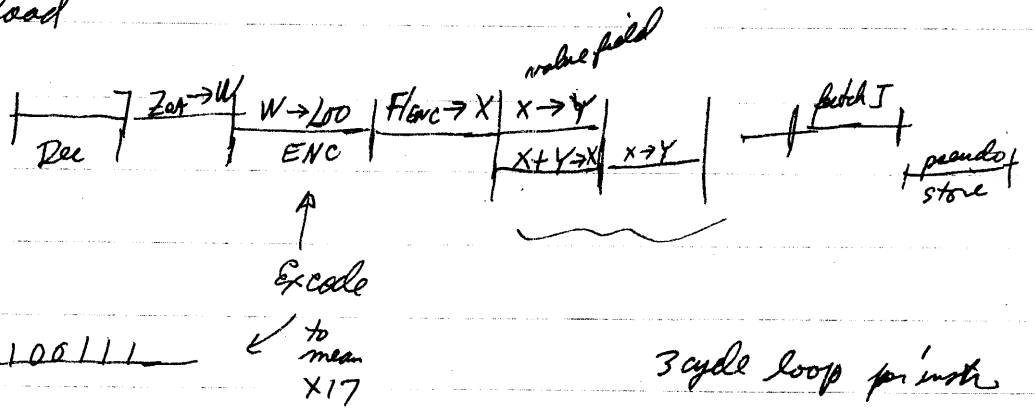


Into LA



7.638

Geo load

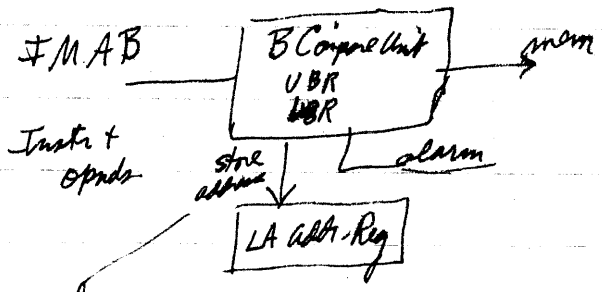


I Box instr uses one of Y regs. may destroy one of preaccessed instrs (The one it was using)

~~fetch~~ fetch from W only one position
if it were Z would have to be either side.

~~Interrupt~~

Memory:
OA
EM >31
XS 16-31
IR 0-15
EIR external internal

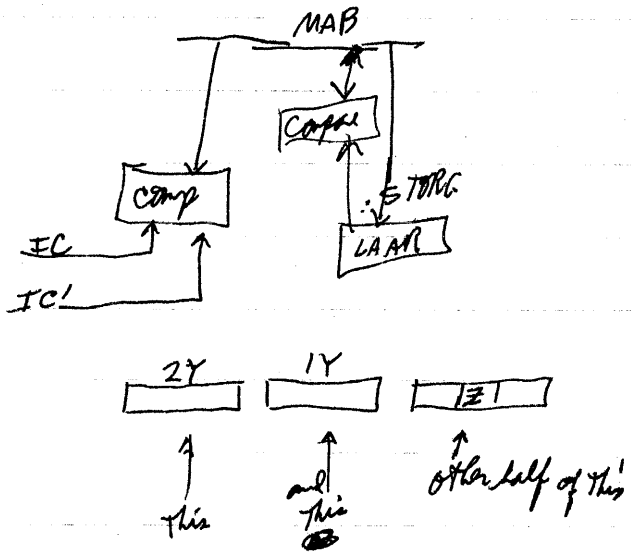


compared here on going to LA.

~~Sets~~ DS bit — causes "no op" of instr

if Look-ahead full
Z_L can't be emptied - ZR can continue but can't be unloaded.

"Prog. store test" storing into next instr — (or near-by instrs)
or within 3 instrs.



If compare: reset instr. counter & reset - & fetch all over again.

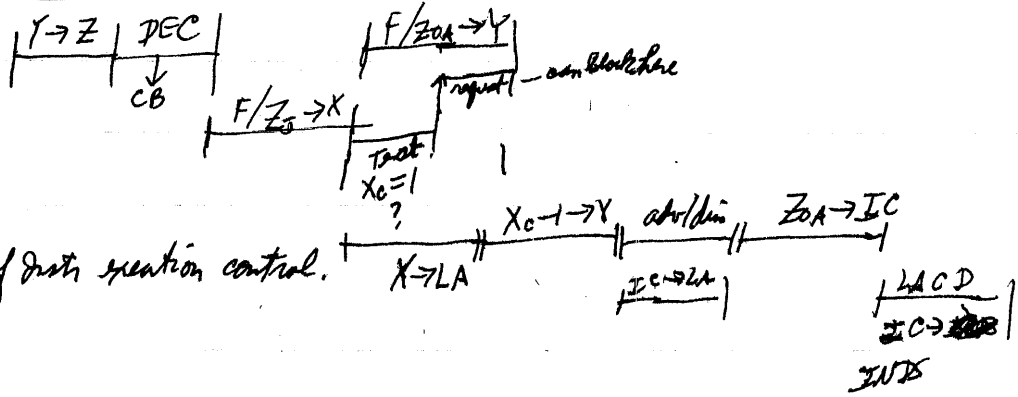
MAB mem address bus
MOB mem out bus.
I CHK BUS
{ LA load lines — not data
IBMAB —

EAR = LAAR effective address. Register

Transfer from I checker to E box 0,4 μs - 0.2 μs if overlapped

decode	Transfer by LA.
0.2	0.2

count + branch : add to BCO - if branch



part of instr creation control.

---

unwanted fetches are kept track of by mem device which keeps track of fetches - reset when branch occurs.

I/O instr : go to LA - Then B exchange.