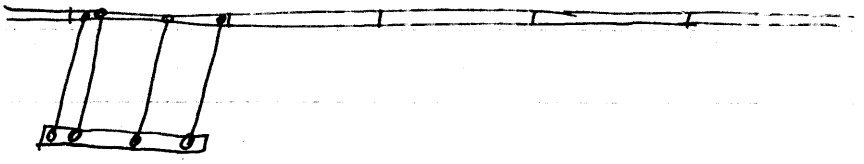
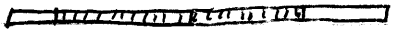
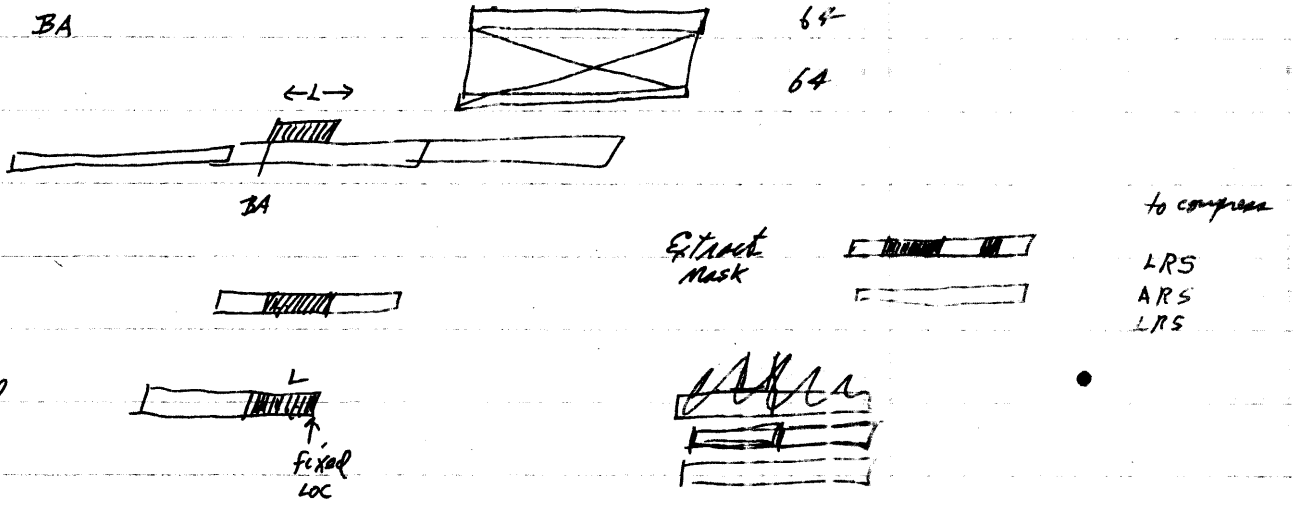
1. Fl Pt. arithmetic
2. Logic.
3. Fixed Point arith
4. Input output







1. Extract or insert a variable length piece of info.
2. Extract several pieces of info with gaps between
3. To be able to assemble or disassemble logical pieces
4. to be able to do other logical connectives
5. to be able to do integer arithmetic on fixed pt fields.

STRETCH.

704

to extract (or insert)

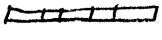
1. Load (OSTNT): Mask	3.0 μs	
2. Connect to acc: Mem (to mem)	3.0 μs	

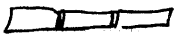
can cross word boundaries
can off-set variable amt.

1. Load Mask (store Mask)	24 μs	
2. and to acc. (and to mem.)	48 μs	
with offset 3. shift R or L before store	~24 μs	

within word only.

To assemble info, with gaps.

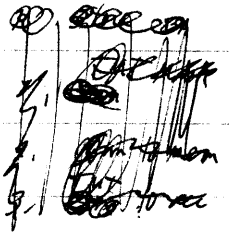


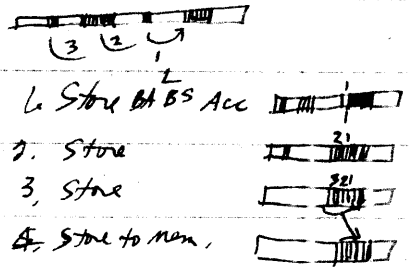


equal groups can be handled by "byte size" Technique done in serial

1. STORE BA BS L (up to 1 word) ← more than one wd needs another

unequal groups must be done in pieces





1. Store BA BS Acc
2. Store
3. Store
4. Store to mem.

can index

can store across wd. boundaries

equal groups or unequal groups.

1. LR
2. ALS
3. LRS
4. ALS
5. STO

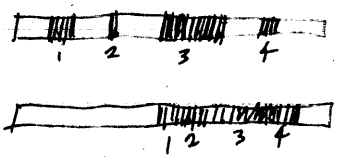
done in serial
4 8 μs / byte

up to 1 wd, more than one needs another

can index

1. ability to fetch fields across wd. boundaries is desirable
2. Field length specifiable is good for fetch + store (extract + insert)
3. offset can be replaced by shift order.
4.

would like to fetch & pack in acc (or store pieces in mem)

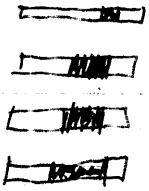


rest undisturbed

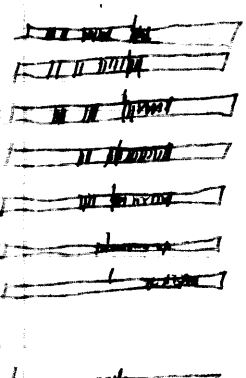
24
8 3
Time = 192

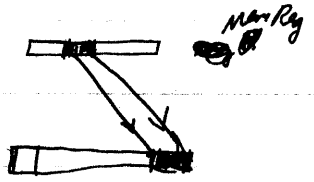
Time ≐ 10

1. conn (A only) BA L off to acc
2. "
3. "
4. "



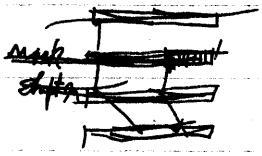
1. LRS
2. ARS
3. LRS
4. ARS
5. LRS
6. ARS
7. LRS
8. sto x
9. cla Mask
10. add to acc
11. or to acc x

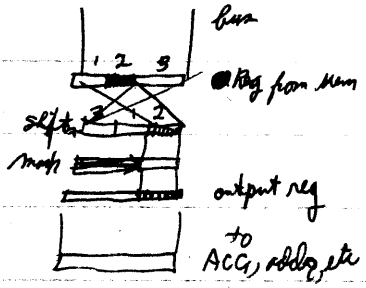




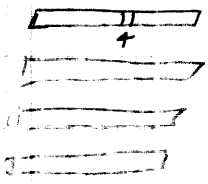
A to B

Mask reg.



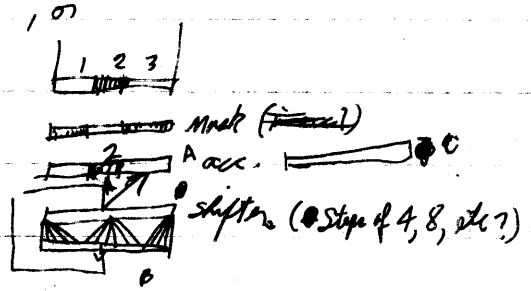


1 conn L BS
2 shift
3. conn
4. shift
5. conn
6 shift
7. conn
8. shift
9. conn to mem

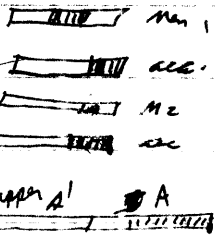


to do in parallel, or

1. connections
2. shifter



(Load) BA, L
add BA2, L
(req parallel adder used)



mpy result double length.