IBM CONFIDENTIAL

Date:	May 12, 1967
From (location S. mail address):	Advanced Computing Systems Menlo Park
& Bldg.:	986/031
Telephone Ext.:	

IBM

Subject:	Architecturally Critical Paths in the MPM
Reference:	
To:	Dr. H. Schorr

Attached is a list of critical timing paths within the MPM from an architectural point of view. Degradation in any of these paths would have a major detrimental effect on overall MPM performance. By overall is meant a global effect, rather than a local effect such as slippage in divider performance. Of the twelve points noted, those involving the contender stacks and interlocking are by far the most critical.

E H Sussenguth

E. H. Sussenguth

EHS:slb

cc: SADL

054

I. Effective address path: (7 cycle path)

ea generation (three input add)	1
bus to BLCU	1/2
BLCU interference resolution	1
storage delay including bussing	3
BLCU decision per tag entry	1
bus to MPM	1/2
internal MPM bus to functional unit	0

II. By-pass from functional unit output to input (0 cycle path)

1. Full bypassing is eminently desirable.
2. If specialized bypassing is necessary the following groupings are the most important:

   add to add
   add to mpy
   mpy to add
   mpy to mpy
   add to cmp
   mpy to cmp

   mixed mpy to d. p. add
   d. p. add to d. p. add

   integer add (with respect to carry register)

   shift to shift
   shift to logic
   logic to shift
   logic to logic
   shift to cmp
   logic to cmp

   index add to ea add
   index add to cmp

   cmp to branch/skip control

III. A-unit interlock control

When an instruction satisfies its interlock constraints, it must be logically removed from contention so that other instructions dependent on it (because of destination-source interlocks or bus conflict interlocks, for example) can start execution on the next cycle.

IV. X-unit interlock control

When an X-contender stack position is vacated, it is refilled with another instruction so that the new instruction can be interlocked and vacated on the next cycle.

The X-unit register data is bussed to the functional units simultaneously with the interlock determination. If the interlocks fail, the functional unit action is logically stopped in such a way that it can restart on the next cycle. (In particular, a unit with a pipeline rate of 2 or more, must not be "busy" working on the illegitimate data.)

V. Instruction start-up path (3 cycle path in X-unit)

Storage bus to IB's	0 (bypass to dispatcher?)
IB to dispatch register	1
Dispatch to contender	1
Contender to functional unit	1 (2 in A-unit)

VI. Effective branch address path

The worst case timing situation occurs when an EXIT has been detected (in the X-dispatch registers) and the BRANCH instruction has not been executed (is in the X-contender stack).

The computation path is:

```
interlock tests on BRANCH                  cycle 1
compute eba, successful/unsuccessful       cycle 2
test top DO table entry:
    if DO entry is correct:
        next instructions to dispatchers   cycle 3
    if DO entry is incorrect:
        correct DO table                   cycle 3
        next instructions to dispatchers   cycle 4
```

VII. DO Table alteration

On each cycle both A- and X-pointers can be moved, an old entry be deleted, and a new entry be accepted.

056

VIII. DO table control of instruction flow

The table entries indicate the number of cycles required to validate DO table entries and permit movement of new instructions to dispatch registers.

if top DO entry is	correct	incorrect	
if required instructions in	IB	IB	storage
unsuccessful branch exit	1	2	1 + access
successful branch exit	1	2	1 + access
no exit (normal sequence)	1	2*	1 + access*

*pathological case (hence unimportant)

IX. Next-fetch mechanism

On each cycle the next-fetch mechanism must search IB addresses, send an address to BLCU, search PSC registers, increment its contents by 8, and accept an override signal from the branch control.

X. Computation dependent SKIPs

The following sequence of instructions illustrates the problem

$A^3 \leftarrow$ any A instruction
$C_2 \leftarrow A^3 \geq A^{10}$
SKIP if C_2 or C_{30}
* any A instruction

The data/control sequence is

end of computation (A-unit)	cycle 1
result to compare unit (A-unit)	cycle 2
compare result to condition bit	
condition bits to skip test unit	cycle 3
compute skip condition (X-unit)	
skip condition to A-unit interlocks	
start bussing on NOP the *-ed op (A-unit)	cycle 4

057

The sequence noted (A-unit compare, SKIP, * on A-op) is probably the worst case as the path involves A-to-X and X-to-A communication and is a relatively frequent occurrence in code. The dual sequences are:

(X-cmp, SK, * on A):	X-unit skew should alleviate this
(X-cmp, SK, * on X):	no inter-unit paths (but important in X-unit)
(A-cmp, SK, * on X):	one inter-unit path, of less programming significance

XI. Computation dependent branches

A discussion similar to VIII obtains. An illustrative sequence is:

$A^3 \leftarrow$ any A instruction

$C_2 \leftarrow A^3 \geq A^{10}$

BRANCH if C_2 or C_{30}

EXIT

XII. Functional unit performance

The current performance of the functional units are noted below

Floating point, 48-bit	ADD	3/1 and 4/1
	MPY	3/1
	DIV	10/7 or 10/8
	CMP	1/1
Floating point, 96-bit	ADD	4/1
	MPY	5/3
	DIV	17/14
	CMP	1/1 (maybe 2/1)
Floating point, mixed	MPY	3/1
	DIV	10/7
Integer	ADD	2/1
	MPY	4/2
Index integers	ADD	1/1
	MPY	4/2 (improve to 3/1)
	DIV	13 max, 8 avg (improve to 8 max)
	CMP	1/1
Shift, logic, moves (A and X)		1/1

058