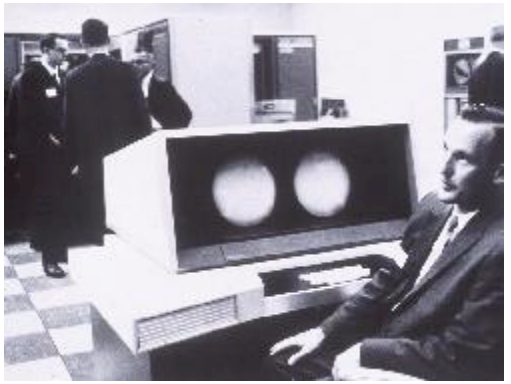
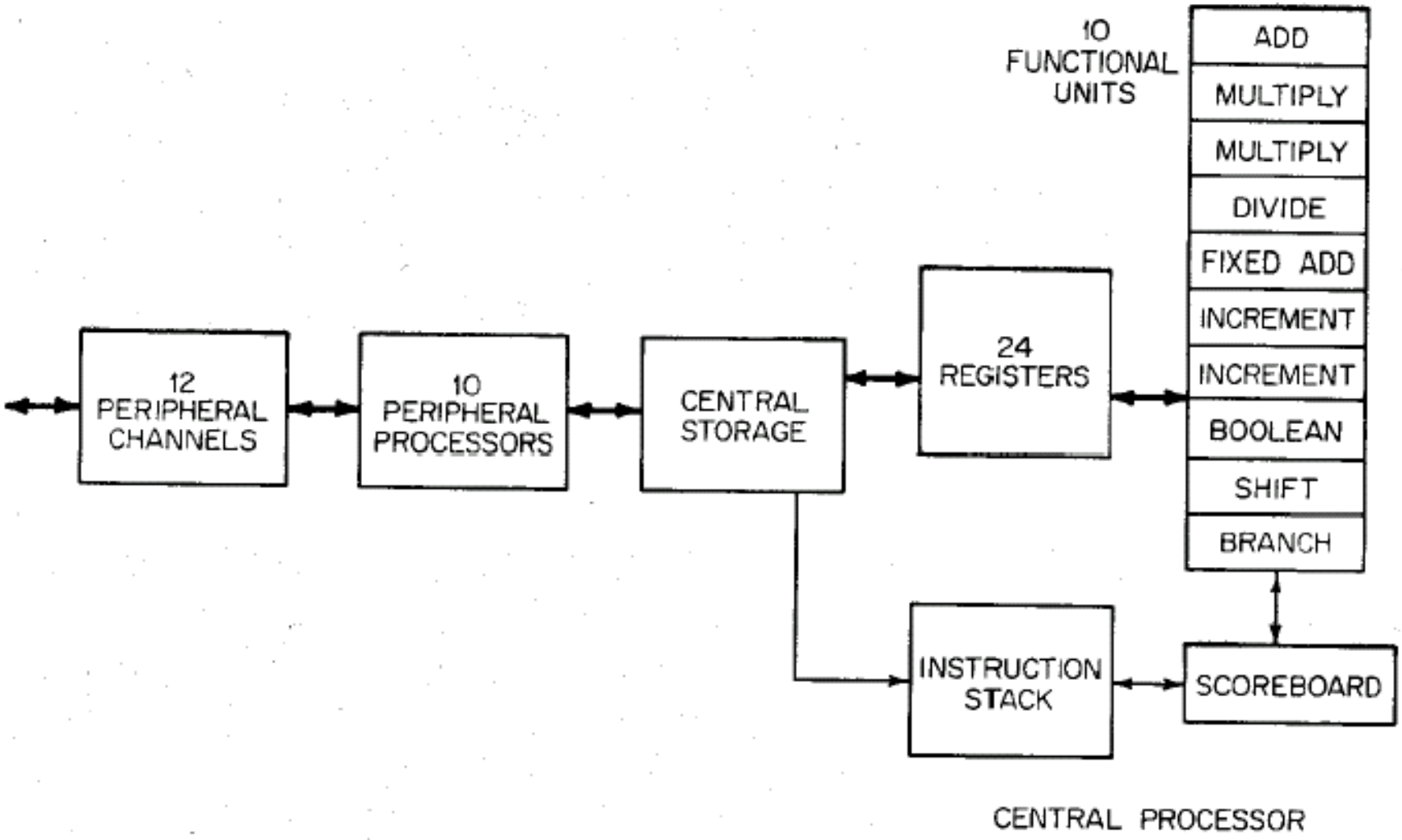
Parallel Operation in the Control Data 6600

Jiun-Hung Chen
Adrienne Wang

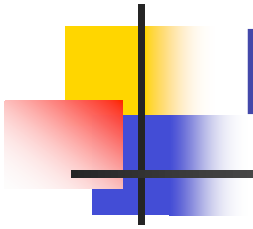
Jan 19, 2005



Overall System



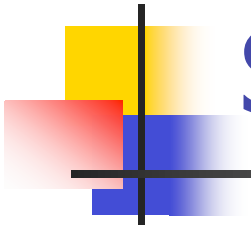
Instruction Level Parallelism in Hardware



- CDC6600: Out-of-order execution -> out-of-order completion
- Multiple functional units: can have multiple instructions in execution phase

Conflicts and Solutions

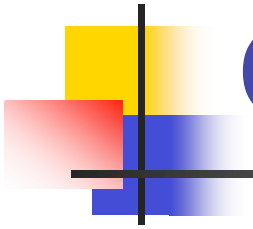
- First Order Conflict (Structural Hazard)
  - Instructions require the same functional units/result registers.
  - Solution: Determine early and issue the 2nd instruction upon completion of the 1st one. Or provide two units to reduce the probability of conflict.
- Second Order Conflict (RAW Hazard)
  - Instructions require results that are not ready.
  - Solution: Scoreboard control over the functional unit.
- Third Order Conflict (WAR Hazard)
  - Some instructions may finish earlier than previously issued instructions and need to overwrite the value in a register which is still needed.
  - Solution: Hold the result in the functional unit.



Scoreboard

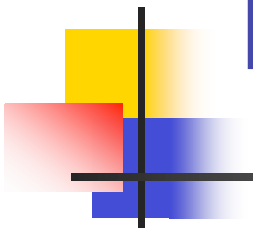
- Also called Unit and Register Reservation Control
- The Scoreboard manages simultaneous operation of multiple functional units on a single instruction stream.
- Units proceed independently.
- The Scoreboard determines when a functional unit can read the operands and write to the result register.

Four Stages of Scoreboard Control



- **1.Issue — decode instructions & check for structural hazards**
- **2.Read operands — wait until no data hazards, then read operands**
- **3.Execution — The functional unit begins execution upon receiving operands. It notifies the scoreboard when it has completed execution.**
- **4. Write result — Write the result to register after the scoreboard sends signal to the functional unit.**

Functional Unit Status



- fields for each functional unit
  - Fm: Function to be performed (eg. + or -)
  - Fi: Destination register
  - Fj, Fk: Source registers
  - Qj, Qk: Functional units producing Fj, Fk
  - Rj, Rk: single-bit flag indicating when Fj, Fk are ready
  - Xi: Identifies which unit has reserved register Xi for its result (Some units may have Bi/Ai)

Scoreboard Operation I

- Place Reservations
  1. Set the unit busy: Prevent the next instruction which uses the same functional unit from being issued.
  2. Set the register designators Fi, Fj, Fk: Transfers the i, j, k fields of the instruction to the designators of the functional unit.
  3. Set the functional unit identifiers Qj, Qk: Copy from the X/B/A identifiers.
  4. Assign the functional unit number to the result register identifier, Xi, Bi, or Ai.

Scoreboard Example Cycle1

Instruction	i j k	Issue	Read	Execute	Write
LD	F6 34+ R2	1			
LD	F2 45+ R3				
MULT	F0 F2 F4				
SUBD	F8 F6 F2				

Functional Unit	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	LD	F6		R2				Yes
Mult1	No								
Add	No								

Scoreboard Example Cycle7

Instruction	i j k	Issue	Read	Execute	Write
LD	F6 34+ R2	1	2	3	4
LD	F2 45+ R3	5	6	7	
MULT	F0 F2 F4	6			
SUBD	F8 F6 F2	7			

Functional Unit	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F2		R3				Yes
Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
Add	Yes	Sub	F8	F6	F2		Integer	Yes	No

Scoreboard Operation II

- Set read flags Rj, Rk
  - If both are set, the unit may start
  - Determined by the Qj, Qk identifiers and by the Release signal from the functional units identified by Qj, Qk. Solve the second order conflict.
  - x6 = x1+x2 Add unit

    x7 = x5/x6 Divide unit

    x6 is the result of the Add unit

    Qk=17

    Rk is determined by Qk and the release signal from the Add unit.

Set the read flags

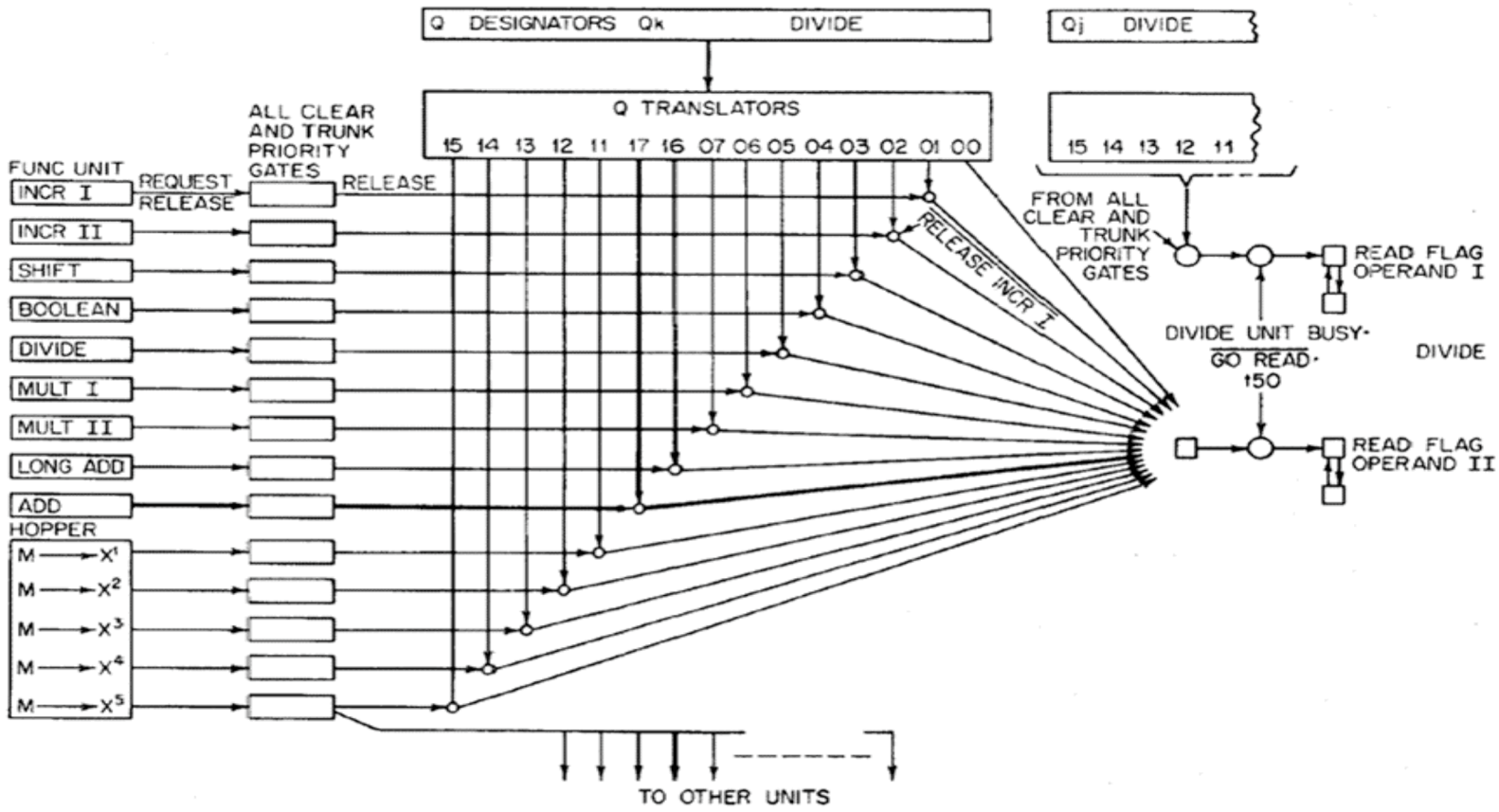
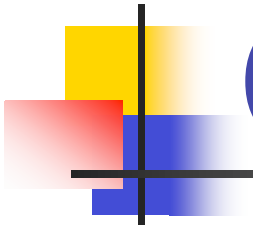


FIGURE 79 Set read flags divide unit.

Scoreboard Operation II (continued)

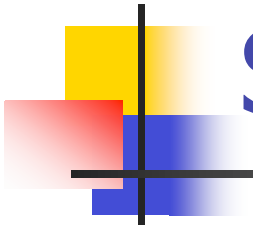


- Send the Release signal to functional units.
- Release the result to the result register.
- Solve the third order conflict.
- x5 = x4 * x3 Mult unit

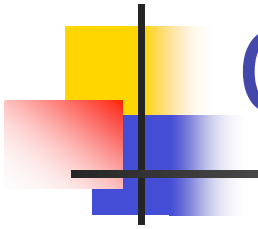
  x4 = x0 + x6 Add unit

  The Add unit can't release the result to x4 unless the Mult unit has read the value in x4.

Limitations of CDC6600 Scoreboard



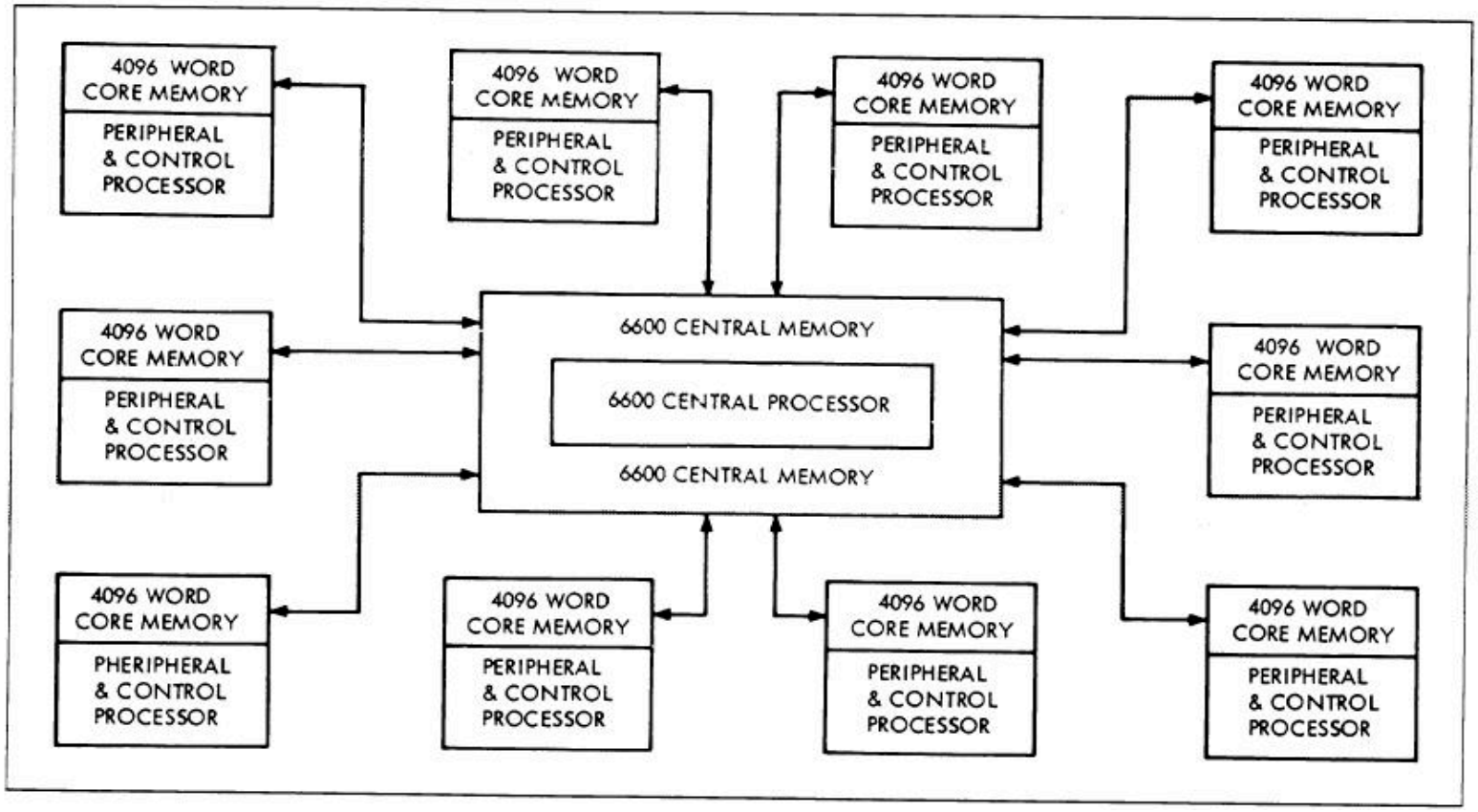
- No forwarding hardware
- Whether independent instructions can be found to execute.
- Limited to the size of the scoreboard.
- Small number of functional units leads to structural hazards
- Antidependences and output dependences lead to WAR and WAW stalls



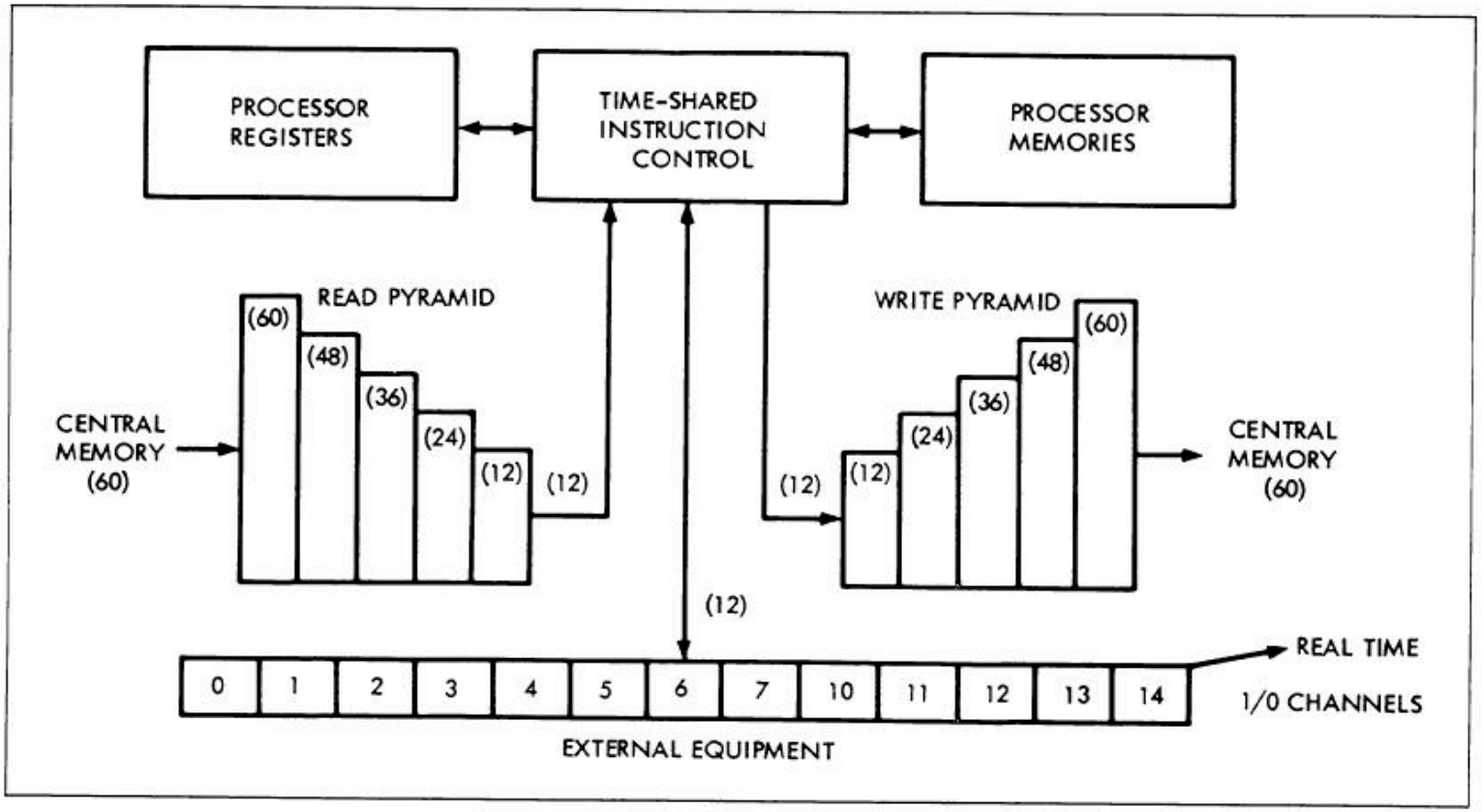
Questions

- Why are all functions separated into different units? Wouldn't it work better if any unit could perform any operation, reducing conflicts of function type?
- A harder problem would be organizing the instructions for best performance and least conflict. Neither paper talks about this possibility.
- How Q is notified the operation is done?
- Any computational theory is able to model the ILP?
- Does the scheduling and communication overhead make this impractical (slow)?
- This seems to be the hardware approach to introducing ILP, could a compiler that was aware of how it was working help the hardware out? How?
- What exactly is a major and minor cycle? I get that the first is longer…
- Is this along the lines of what current processors do? How do they stay synched and consistent?
- Seems to be substantial bookkeeping. Performance evaluation? Overhead?
- Multiple memory units seem good for speed. Don't they have dependency problems?
- Are any modern techniques inspired by scoreboard?

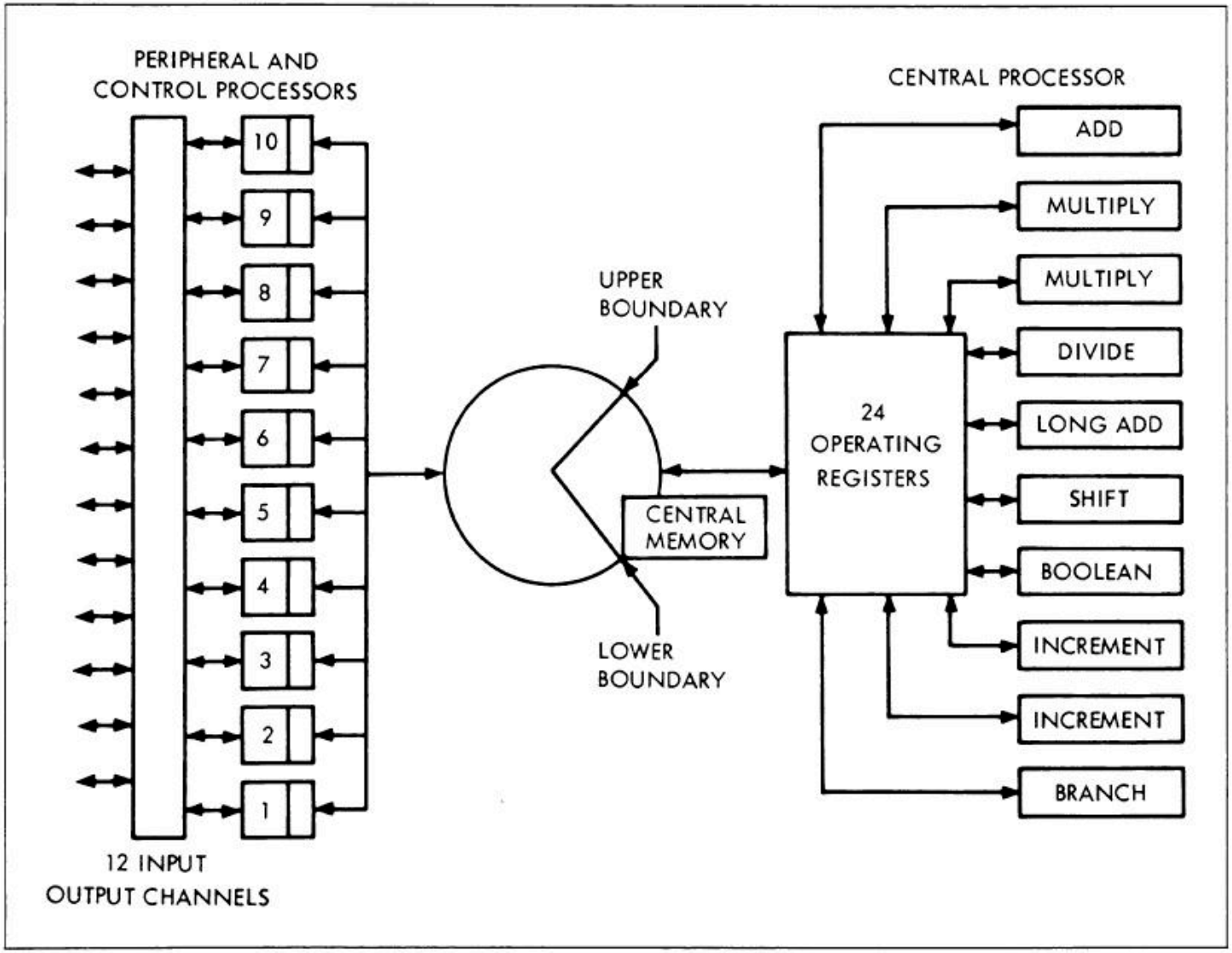
Peripheral and Central processors



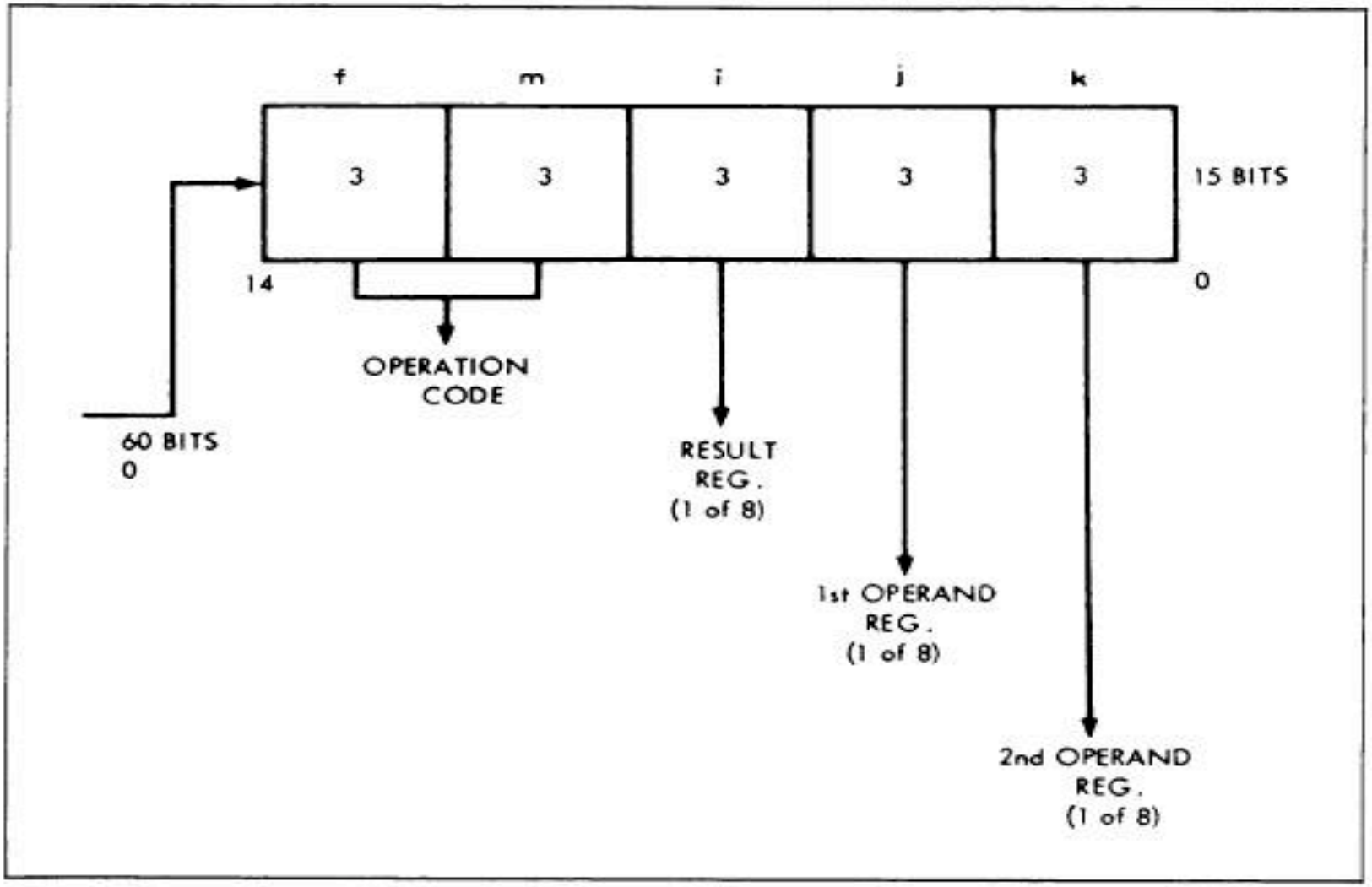
Time-Sharing Design



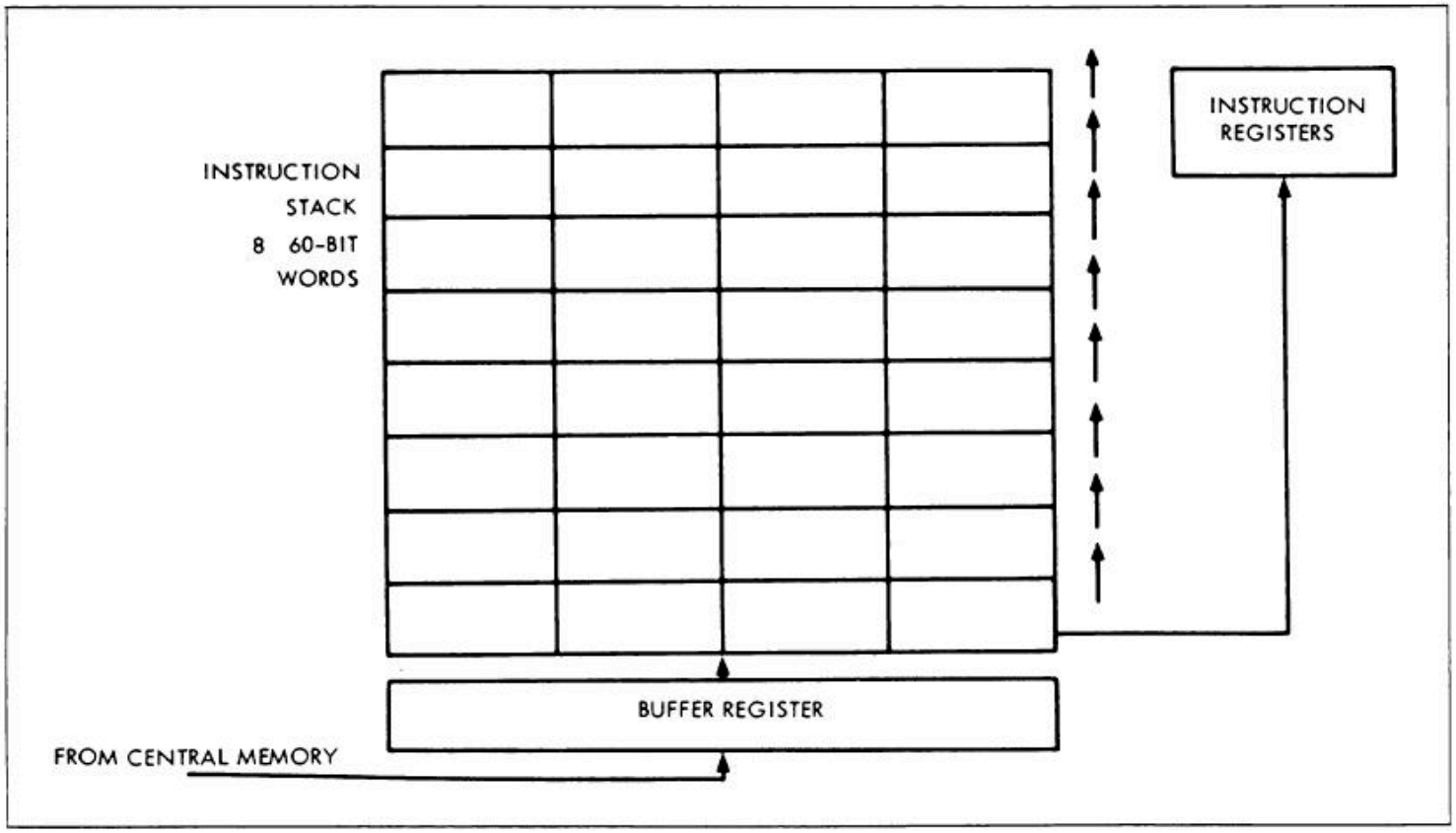
Central Processor



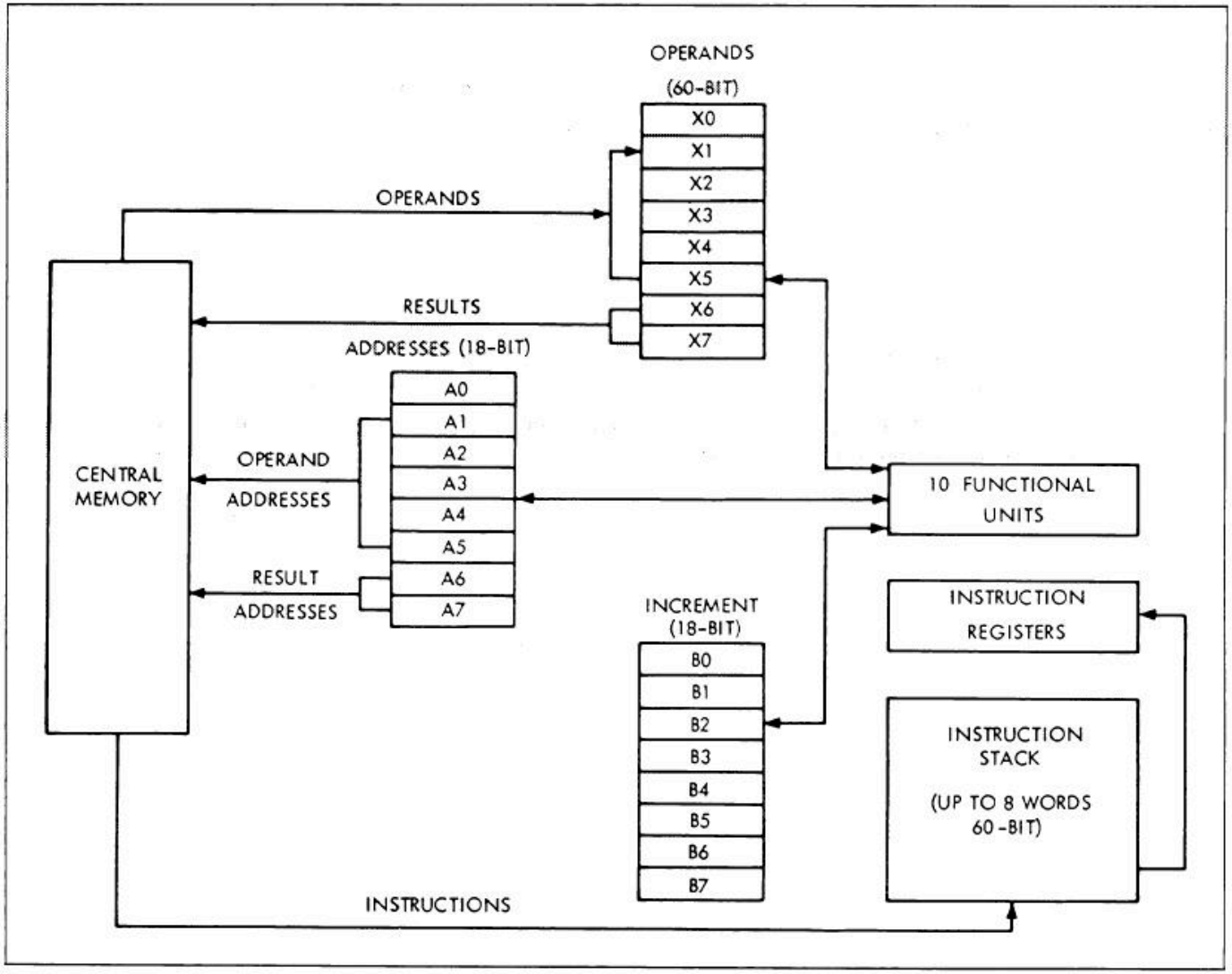
Instruction Format

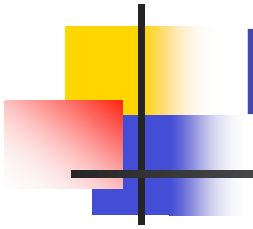


Instruction Stack



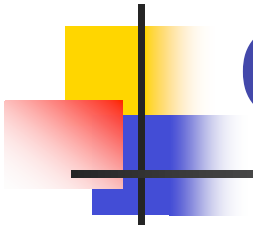
Central Processor Operating Registers





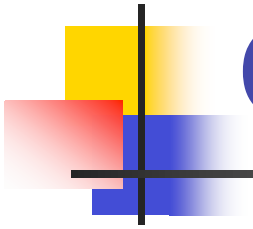
Features

- Parallel Operations
- Scoreboard
- All-transistor Logic



Critique

- Pros
  - Dynamic Scheduling with Scoreboard
- Cons
  - Large number of buses
  - Multiple function units



Question

- Performance vs. number of function units
- Scoreboard vs. Tomasulo?

Scoreboard Example

Instruction	i j k	Issue	Read	Execute	Write
LD	F6 34+ R2				
LD	F2 45+ R3				
MULT	F0 F2 F4				
SUBD	F8 F6 F2				

Functional Unit	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer									
Mult1									
Add									

Scoreboard Example Cycle1

Instruction	i j k	Issue	Read	Execute	Write
LD	F6 34+ R2	1			
LD	F2 45+ R3				
MULT	F0 F2 F4				
SUBD	F8 F6 F2				

Functional Unit	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	LD	F6		R2				Yes
Mult1	No								
Add	No								

Scoreboard Example Cycle2

Instruction	i j k	Issue	Read	Execute	Write
LD	F6 34+ R2	1	2		
LD	F2 45+ R3				
MULT	F0 F2 F4				
SUBD	F8 F6 F2				

Functional Unit	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	LD	F6		R2				Yes
Mult1	No								
Add	No								

Scoreboard Example Cycle3

Instruction	i j k	Issue	Read	Execute	Write
LD	F6 34+ R2	1	2	3	
LD	F2 45+ R3				
MULT	F0 F2 F4				
SUBD	F8 F6 F2				

Functional Unit	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	LD	F6		R2				Yes
Mult1	No								
Add	No								

Scoreboard Example Cycle4

Instruction	i j k	Issue	Read	Execute	Write
LD	F6 34+ R2	1	2	3	4
LD	F2 45+ R3				
MULT	F0 F2 F4				
SUBD	F8 F6 F2				

Functional Unit	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	LD	F6		R2				Yes
Mult1	No								
Add	No								

Scoreboard Example Cycle5

Instruction	i j k	Issue	Read	Execute	Write
LD	F6 34+ R2	1	2	3	4
LD	F2 45+ R3	5			
MULT	F0 F2 F4				
SUBD	F8 F6 F2				

Functional Unit	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	LD	F2		R3				Yes
Mult1	No								
Add	No								

Scoreboard Example Cycle6

Instruction	i j k	Issue	Read	Execute	Write
LD	F6 34+ R2	1	2	3	4
LD	F2 45+ R3	5	6		
MULT	F0 F2 F4	6			
SUBD	F8 F6 F2				

Functional Unit	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F2		R3				Yes
Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
Add	No								

Scoreboard Example Cycle7

Instruction	i j k	Issue	Read	Execute	Write
LD	F6 34+ R2	1	2	3	4
LD	F2 45+ R3	5	6	7	
MULT	F0 F2 F4	6			
SUBD	F8 F6 F2	7			

Functional Unit	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F2		R3				Yes
Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
Add	Yes	Sub	F8	F6	F2		Integer	Yes	No

Scoreboard Example Cycle8

Instruction	i j k	Issue	Read	Execute	Write
LD	F6 34+ R2	1	2	3	4
LD	F2 45+ R3	5	6	7	
MULT	F0 F2 F4	6			
SUBD	F8 F6 F2	7			

Functional Unit	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer	Yes	Load	F2		R3				Yes
Mult1	Yes	Mult	F0	F2	F4	Integer		No	Yes
Add	Yes	Sub	F8	F6	F2		Integer	Yes	No