UAH

Pipelining-0

590005
Arquitectura e Ingeniería de Computadores

Goal

To transform a Von-Neumann organization into a pipelined microarchitecture

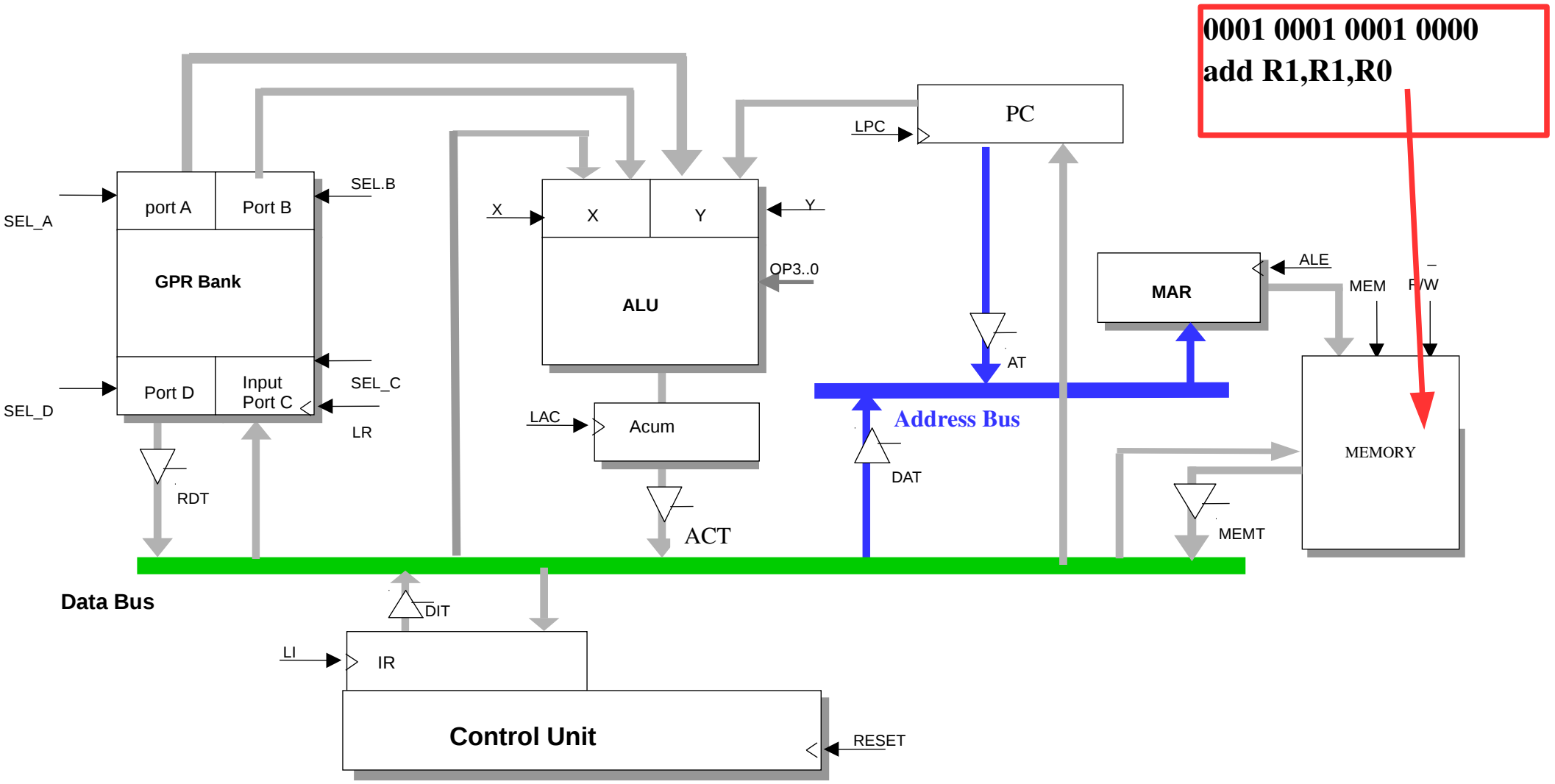
Transformación de una organización Von-Neumann en un cauce segmentado

Fases

- 1. Búsqueda de la instrucción - Instruction FETCH (IF)
  - RI <---M(PC) .
- 2. Incremento PC (IPC)
  - PC ← PC ++
- 3. Decodificación – Instruction Decoding (ID)
  - Register RI contains the instruction
- 4. Ejecución (ej: instr de proceso like ADD or SUB or SHL)
  - 4. Búsqueda de operandos -Operands Fetch (OF)
  - 5. Ejecución de la operación -Operation EXecution (EX)
  - 6. Almacenamiento del resultado -Store Result (in GPR: WB)

UAH

Von Neumann



Fetch

- Steps to read a memory word
- from a location pointed to by the Program Counter
- cycle by cycle, specifying active signals in each cycle

- Definir paso a paso la lectura de una palabra en memoria
- almacenada en una posición apuntada por el PC
- especificando ciclo a ciclo las señales activas necesarias

Crono-tabla

- Use a table layout as shown and complete the full execution of instruction ADD R1,R1,R0
- Assume memory latency is only 1 cicle (Cache)

Transforms

- Adder to increment the PC off the ALU
- Overlap Increment PC phase with Fetch Phase
- Maximize overlapping to obtain min. CPI (cycles per instr.)

- Añadir un sumador de uso exclusivo del PC
- Solapar ciclo de incremento de PC con el de Fetch
- Maximizar solapamientos para minimizar el CPI (ciclos por instrucción de proceso)

UAH

Add R1,R1,R0
R1 ← R1+R0

Fase	*Ciclo*	*Op El.*	*Señales*
IF + IPC	1	MAR ← PC	AT,ALE
	2	IR ← M(MAR) PC ← PC++	MEM,R,MEMT,LI INCR,LPC
ID	3		
OF/EX	4	Acum ← R1+R0	SEL_A/B,X,Op,LAC
WB	5	R1 ← Acum	ACT,SEL_C,LR,Reset

$$A = (I \cdot 7 \cdot Tc)/(I \cdot 5 \cdot Tc) = 7/5 = 1{,}4 \quad 40\%$$

UAH

Further Transforms / Más transforms

- If instructions are simple to decode (RISC), may use ID cycle to fetch operands and perform operation
- This may require a longer cycle time

- Si las instrucciones son simples (RISC), no es necesario un ciclo exclusivo de decodificación: asimilamos ID con la operación
- Podría requerir mayor tiempo de ciclo

UAH

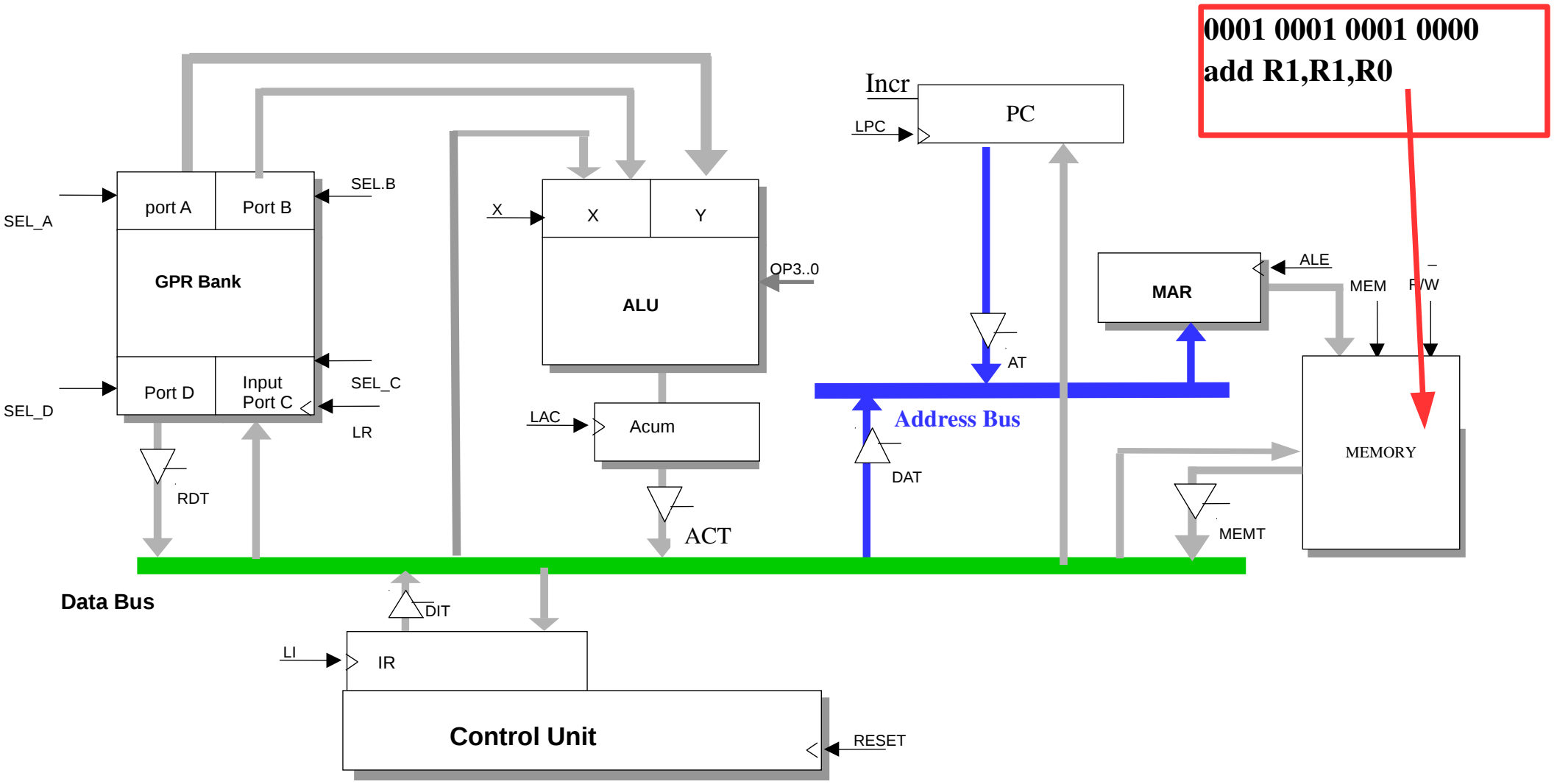
Add R1,R1,R0
R1 ← R1+R0

Fase	*Ciclo*	*Op El.*	*Señales*
IF + IPC	1	MAR ← PC	AT,ALE
	2	IR ← M(MAR) PC ← PC++	MEM,R,MEMT,LI INCR,LPC
ID/OF/EX	3	Acum ← R1+R0	SEL_A/B,X,Op,LAC
WB	4	R1 ← Acum	ACT,SEL_C,LR,Reset

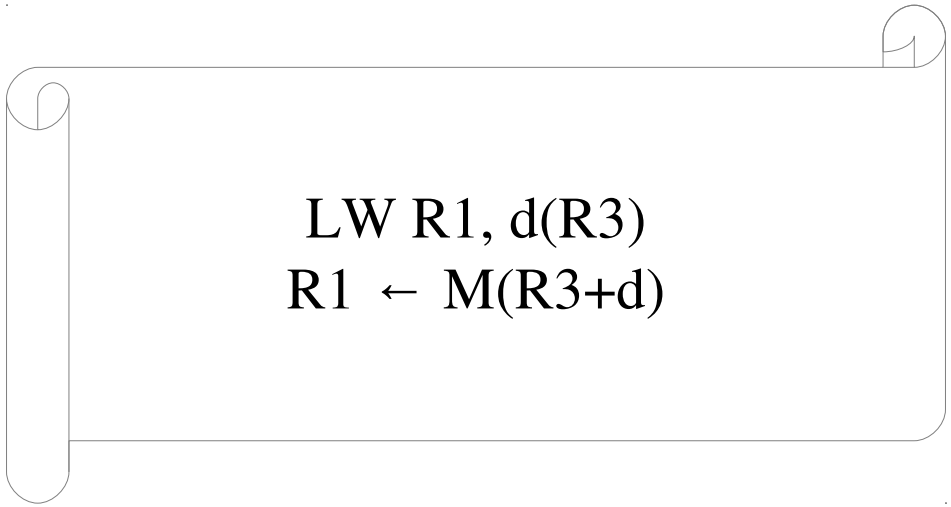
$$A = (I \cdot 7 \cdot Tc)/(I \cdot 4 \cdot Tc) = 7/4 = 1{,}75 \quad 75\%$$

UAH

R-R


0001 0001 0001 0000
add R1,R1,R0
Incr
PC
LPC
port A
Port B
SEL.B
SEL_A
GPR Bank
Port D
Input
Port C
SEL_C
SEL_D
LR
RDT
X
X
Y
ALU
OP3..0
LAC
Acum
ACT
AT
Address Bus
DAT
MAR
ALE
MEM
R/W
MEMORY
MEMT
Data Bus
DIT
LI
IR
Control Unit
RESET

Load Instruction
instrucción de carga



Fases

- 1. IF: RI <---M(PC) .
- 2. IPC: PC ← PC + 4
- 3. ID
- 4. Ejecución (for a LOAD / CARGA)
  - Búsqueda de operandos de dir – Address Operands Fetch (OF)
  - Cálculo de dir. efectiva – Obtain effective address (EX)
  - Acceso a memoria – Memory Access (ME)
  - Almacenamiento en Registro – Store result: Write Bank (WB)

UAH

LW R1, d(R3)
R1 ← M(R3+d)

Fase	*Ciclo*	*Op El.*	*Señales*
IF	1	MAR ← PC	AT,ALE
	2	IR ← M(MAR)	MEM,R,MEMT,LI
IPC	3	Acum ← PC++	Y(),Op(),LAC
	4	PC ← Acum	ACT,LPC
ID	5		
OF/EX	6	Acum ← R3+d	SEL_A,DIT,X/Y,Op,LAC
	7	MAR ← Acum	ACT,DAT,ALE
WB	8	R1 ← M(MAR)	MEM,R,MEMT,SEL_C,LR, *Reset*

UAH

Applying same transforms as before…
Realizando las mismas transformaciones de antes...

LW R1, d(R3)
R1 ← M(R3+d)

Fase	*Ciclo*	*Op El.*	*Señales*
IF + IPC	1	MAR ← PC	AT,ALE
	2	IR ← M(MAR) PC ← PC++	MEM,R,MEMT,LI INCR,LPC
ID/OF/EX	3	Acum ← R3+d	SEL_A/B,X,Op,LAC
	4	MAR ← Acum	ACT, DAT,ALE
ME/WB	5	R1 ← M(MAR)	MEM,R,MEMT,SEL_C,LR, Reset

$$A = (I \cdot 8 \cdot Tc)/(I \cdot 5 \cdot Tc) = 8/5 = 1{,}6 \quad 60\%$$

A CISC instruction
Una instrucción CISC

ADD R1, R4, d+R2(R3++)

R1 ← R4+M(R3+R2+d); R3<--R3+4

UAH

cycles 1& 2 are the same as before

R1 ← R4+M(R3+R2+d);
R3<--R3+4

Fase	*Ciclo*	*Op El.*	*Señales*
ID/OF	3	Acum ← R3+d	SEL_A=R3, DIT, X=Bus,Op=+,LAC
	4	Acum ← Acum+R2	SEL_A=R2,ACT, X=Bus,Op=+,LAC
	5	MAR ← Acum	ACT, DAT, ALE
EX / WB	6	Acum ← M(MAR) +R4	MEM,R,MEMT,SEL_A=R4, X=Bus,Op=+, LAC
	7	R1 ← Acum	ACT, SEL_C=R1, LR
	8	Acum ← R3+4	SEL_A=R3, DIT, X=Bus,Op=+,LAC
	9	R3 ← Acum	ACT, SEL_C=R3, LR, Reset

UAH

Introducing Pipelining

- Have as many instructions in the CPU as execution phases
- Goal: to have CPI=1
- Need to Fetch one instruction each cycle
  - Access Memory every cycle (IF)
  - Whenever there is access to Data in Memory:
  - Structural conflict
- Need simultaneous access to GPRs from all instructions

Identify elements to support those features
Resolve structural conflicts

UAH

Introduciendo Segmentación

- Poder tener tantas instrucciones como fases del cauce
- CPI=1: Fetch de Instrucción cada ciclo
  - Acceso a Memoria en todos los ciclos (IF)
  - Acceso a Memoria de Datos a veces:
  - Conflicto estructural
- Poder acceder simultáneamente a Registros GPR

Identificar qué elementos añadir a estructura
Identificar conflictos estructurales y resolverlos