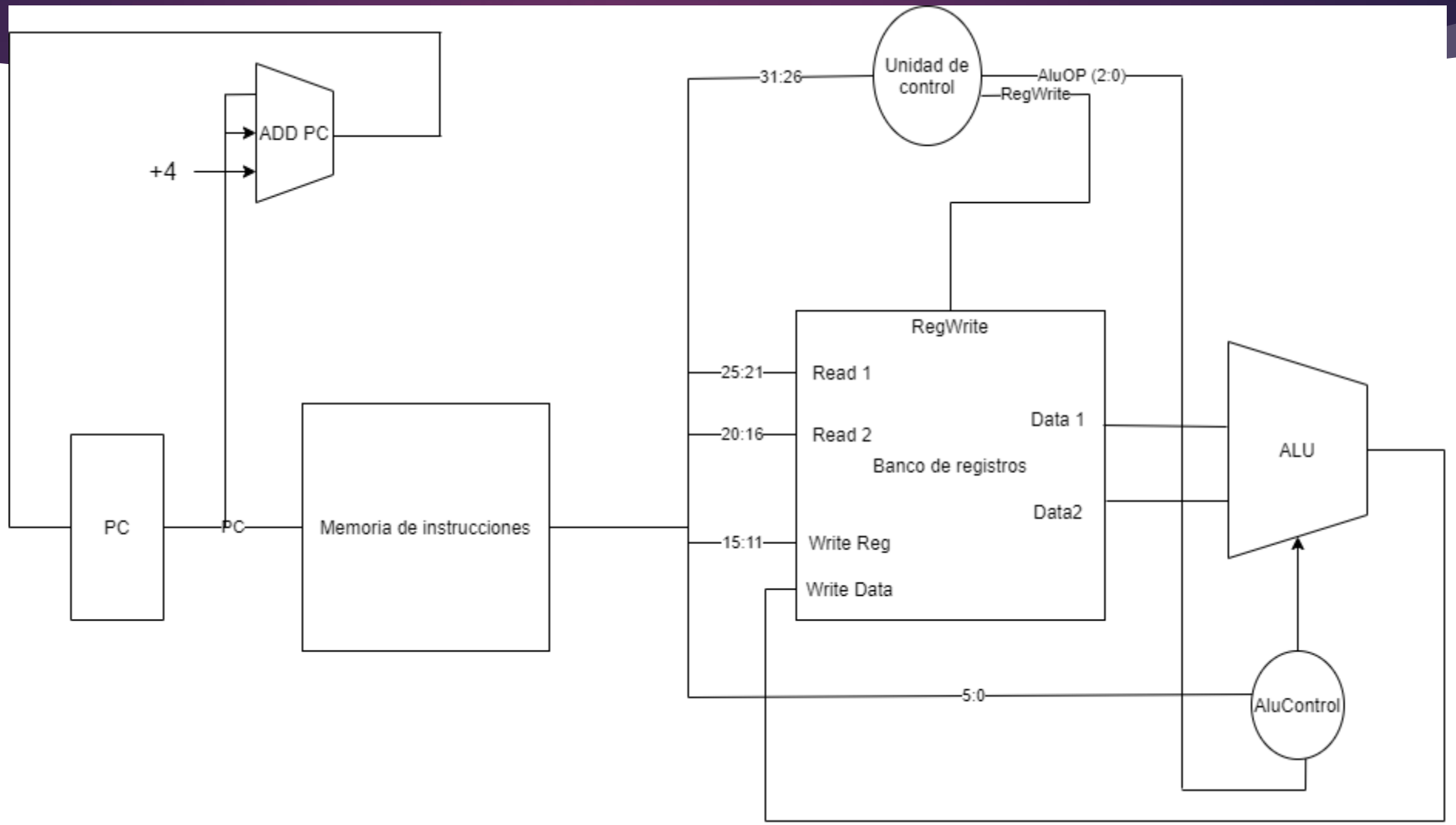
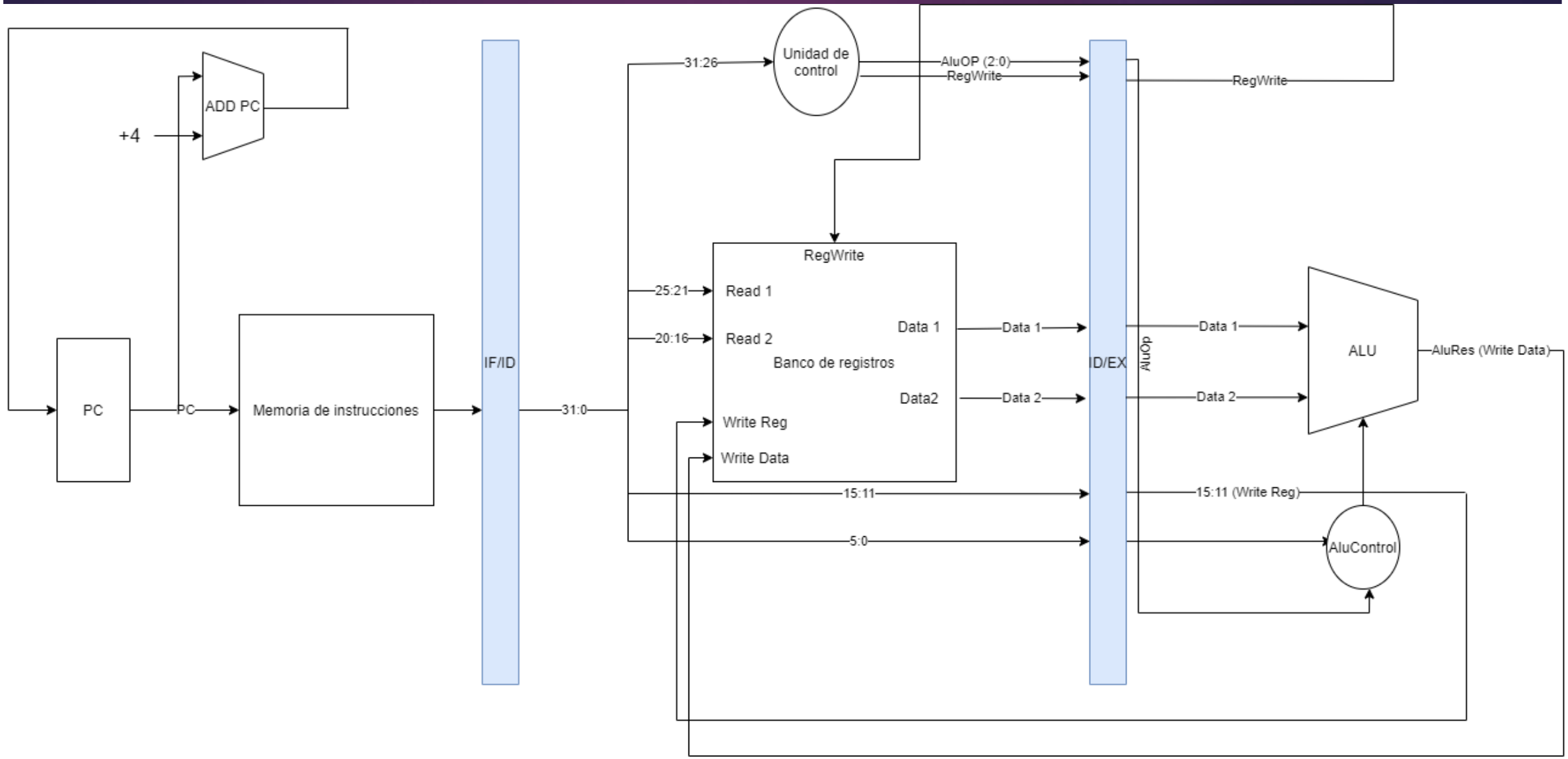
FASE 1 Y 2 (PROYECTO MIPS)

GONZÁLEZ RAMOS JORGE HUMBERTO
OCHOA VELASCO DANA
JIMÉNEZ VITAL FRANCISCO

Datapath Fase 1



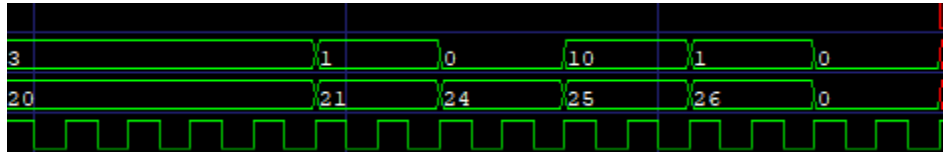

ADD PC
+4
PC
PC
Memoria de instrucciones
IF/ID
31:0
31:26
Unidad de control
AluOP (2:0)
RegWrite
RegWrite
RegWrite
25:21
20:16
Read 1
Read 2
Banco de registros
Data 1
Data2
Write Reg
Write Data
Data 1
Data 2
15:11
5:0
ID/EX
AluOp
Data 1
Data 2
ALU
AluRes (Write Data)
15:11 (Write Reg)
AluControl

Resumen Fase 1

- En esta fase únicamente podemos utilizar instrucciones de tipo R y no contamos con una memoria por lo que no es posible guardar los resultados más que en registros.

Resultados fase 1

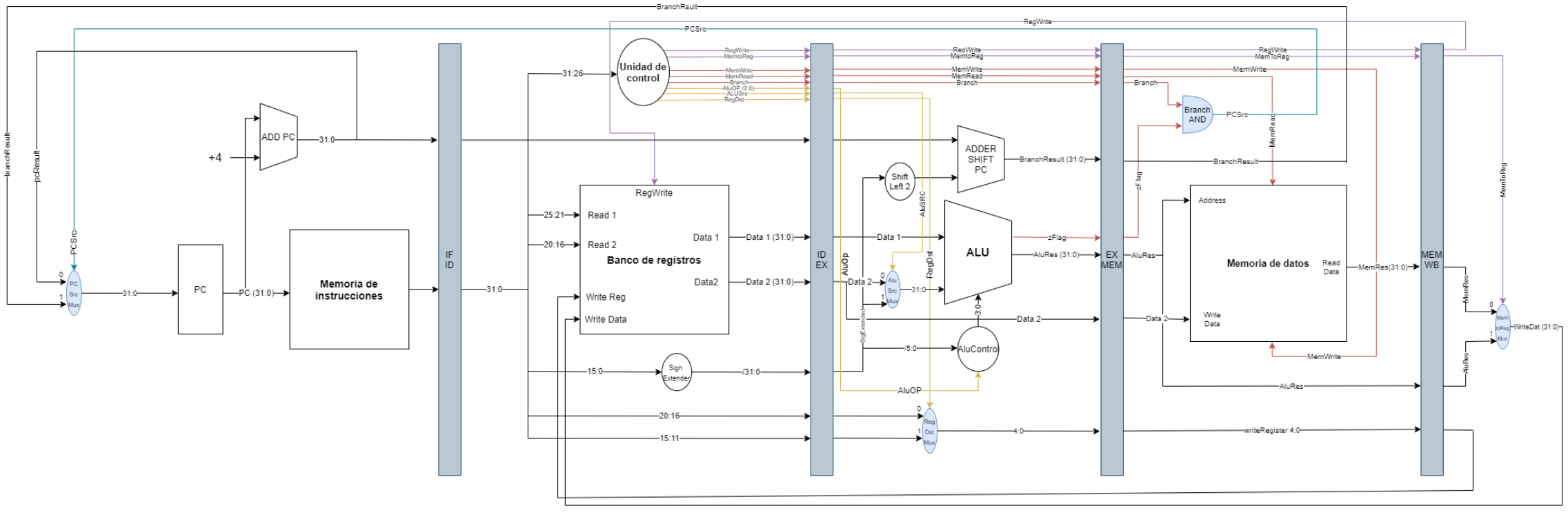
- Se subió unos archivos para probar esta fase y tuvimos estos resultados.



- Los cuales concuerdan con los indicados en moodle

20	3	SUM
21	1	SUB
24	0	AND
25	10	OR
26	1	SLT
0	0	NOP

Datapath fase 2



Resumen fase 2

- En esta fase se añaden módulos para shift, extender de señal y otros módulos como la memoria.
- Con estos módulos es posible crear branches y también hacer uso de la memoria de datos.
- Las instrucciones añadidas son ADDI, SLTI, ANDI, ORI, SW, LW, BEQ.

Resultados fase 2

- Se añadieron algunos módulos de la fase 2, el signal extender y otros módulos (puede verse el avance en el github)

No está terminada al 100% pero en análisis y explicación estará disponible en la wiki al momento de la presentación

Decodificador

- El decodificador ya está terminado, se ha probado con distintos códigos de ensamblador y soporta las siguientes instrucciones.

* NOR, OR, AND, SLT, XOR, ADD, SUB, DIV, MULT, LW, SW,BEQ,J,ADDI,SLTI,ORI,ANDI

Y las siguientes pseudoinstrucciones

* LI,CLEAR, B,BGT,BEG, BEL, BLT

- El decodificador es un archivo de Python el cual terminó con una longitud de 354 en el archivo principal y otro con diccionarios de 91.

Se probaron distintos códigos en ensamblador y fueron convertidos correctamente a instrucciones.

```
main:
    li $t0 10
    li $t1 1

.loop:

    blt $t0 $t1 .end

    addi $t0 $t0 -1

    j .loop

.end:
    li $t0 93
    sw $t0 0($zero)
```

```
main:
    li $t0 10 # Factorial del 5

    # restar 1 a 5 xq blq = a<b

    li $s0 1
    sub $t0 $t0 $s0

    # Continuamos

    li $t1 1 # Numero inicial
    li $t3 1 # factorial
    li $t2 1 # Loop var

.loop:
    blt $t0 $t1 .end

    addi $t1 $t1 1 # $t0 = $t1 + 0
    mult $t3 $t1 $t3

    j .loop
.end:
    sw $t3 0($zero) # guardar factorial en memoria
    lw $t3 0($zero) # Leer
```

Opciones de ensamblador

NÚMEROS PERFECTOS

MÁXIMO COMÚN DIVISOR

DETECTAR EL TIPO DE TRIÁNGULO

SELECTIVE SORT