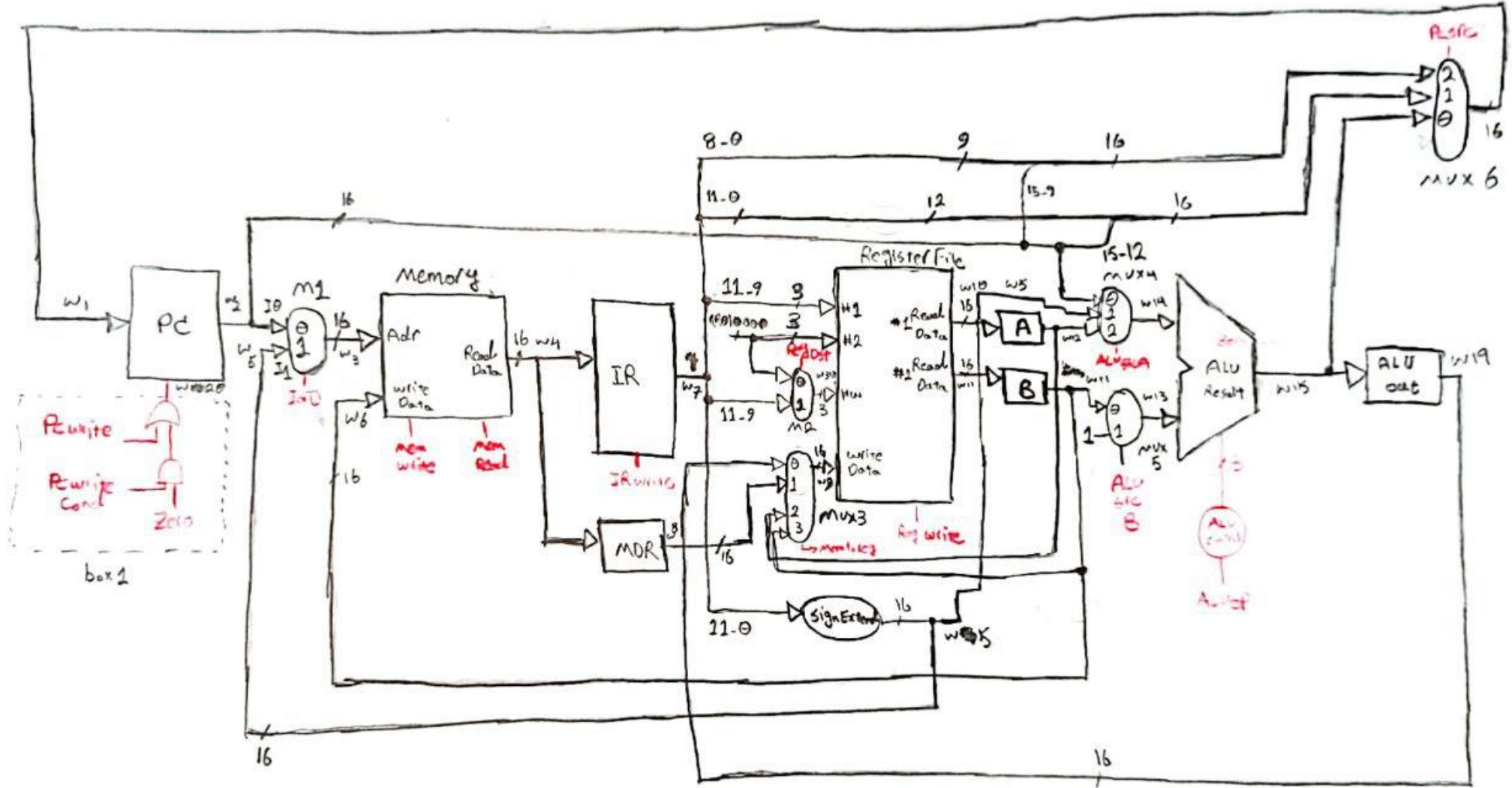# MIPS multicycle processor

Amir Hossein shahgholi 810199441
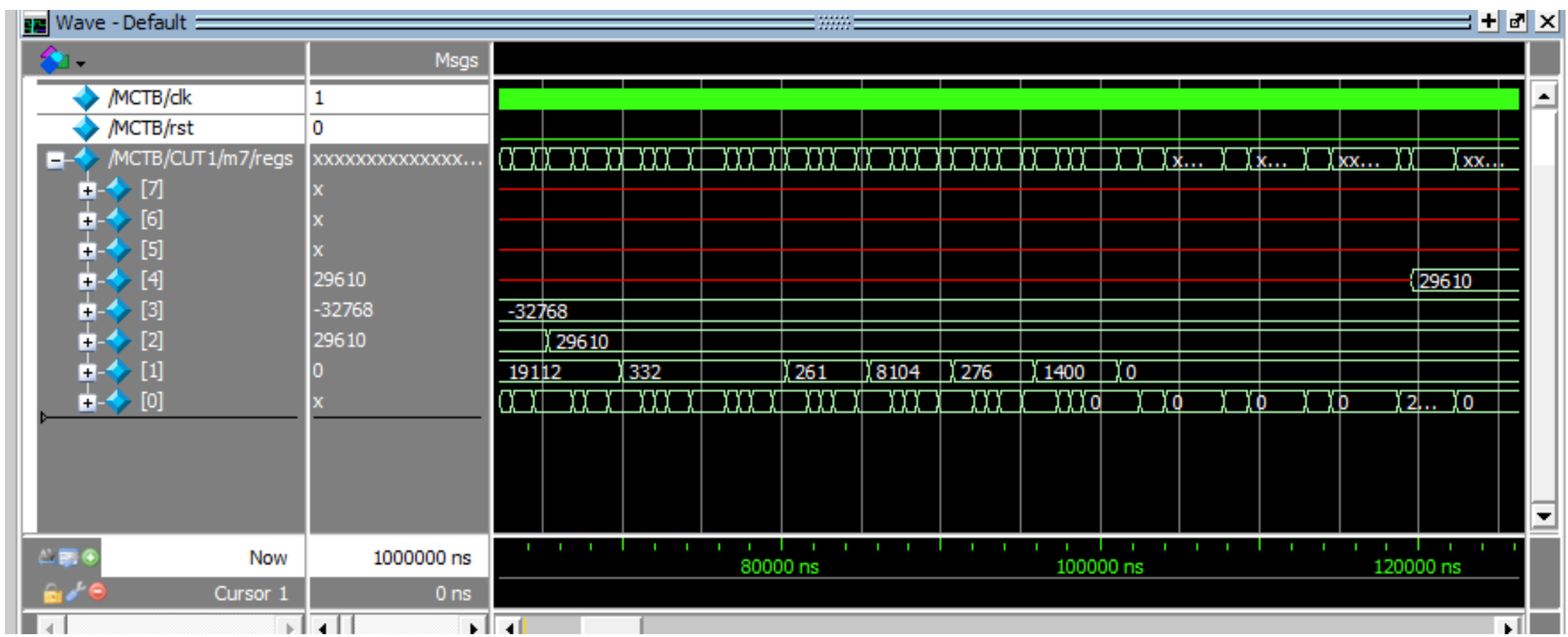
Erfan Soleymani 810199439

Amir Hossein Shahgholi 810199441
Erfan Soleymani 810199439

Finite State Machine (FSM) diagram for a multicycle processor datapath control, with accompanying ALU operation and ALU control tables.

IF state:
IorD = 0
memRead = 1
IRwrite = 1
ALUsrcA = 00
ALUsrcB = 00
PCsrc = 00
Pcwrite = 1
ALUop = 00

ID state:
ALUsrcA = 01
ALUsrcB = 0
ALUop = 00

S7 (1000'1 addi): memtoreg = 00, regDst = 0, regwrite = 1
S1 (1101' subi): ALUsrcA = 01, ALUsrcB = 0, ALUop = 01
S_ (1110 andi): ALUsrcA = 01, ALUsrcB = 0, ALUop = 01
S_ (1111 ori): ALUsrcA = 01, ALUsrcB = 0, ALUop = 01
S5 (0100 J): PCsrc = 01, Pcwrite = 1
S6 (0101 branch): ALUsrcA = 10, ALUsrcB = 0, ALUop = 10, PCsrc = 10, Pcwritecond = 1, branch = 1
S12: Memwrite = 1, IorD = 1
S11: RegDst = 0, memtoreg = 01, Regwrite = 1
S8: memtoreg = 11, Regwrite = 1, RegDst = 1
S10: Memload = 1, IorD = 1
S9: RegDst = 0, Regwrite = 1, memtoreg = 00
S7: ALUsrcA = 10, ALUsrcB = 0, ALUop = 11
Store (0001):
Load (0000):
RT (1000):
RT (func = 000000010): inc
R (func = 000000000):

ALU

ALU op	ALU function
00	Add 010
10	sub 011
01	add 000, sub 001 or 001, and 000
11	and 000, or 001, add 010, sub 011, Not 100, R 101, NoP 111

NoP → out = R0
R → out = Ri

ALUcontrol

01		AluFunction
orc : 1100		010
1101		011
1110		000
1111		001

11		
000000010		101
000000100		101
000001000		011
000010000		000
000100000		001
001000000		100
010000000		111

```
0000000000000000
1100100000000000
1100100000000000
1100100000000000
1100100000000000
1100100000000000
1100100000000000
1100100000000000
1100100000000000
1100100000000000
1100100000000000
1100100000000000
1100100000000000
1100100000000000
1100100000000000
1100100000000000
1000000000010011
0000000100101100
1000000000010001
0000000100101101
1000000000010010
1000000001000001
1000000010000011
0100000011011011
0000000100101110
1000000000010001
0010000000011101
0000000100101110
1000000000010010
1000000000001010
1000000001000001
1000000010000011
0100000100100011
0000000100101111
```

```
0010000001000001
0000000100110010
1000000000010010
1000000000001010
1000000001000001
1000000010000011
0100001001000011
0000000100110011
1000000000010001
0010000001001010
0000000100110011
1000000000010010
1000000000001010
1000000001000001
1000000010000011
0100001010001011
0000000100110100
1000000000010001
0010000001010011
0000000100110100
1000000000010010
1000000000001010
1000000001000001
1000000010000011
0100001011010011
0000000100110101
1000000000010001
0010000001011100
0000000100110101
1000000000010010
1000000000001010
1000000001000001
1000000010000011
0100001100011011
0000000100110110
```

```
0010000001110111
0000000100111000
1000000000010010
1000000000001010
1000000001000001
1000000010000011
0100001111110011
0000000100111001
1000000000010001
0010000010000000
0000000100111001
1000000000010010
1000000000001010
1000000001000001
1000000010000011
0100010000111011
0000000100111001
1000000000010001
0010000010001001
0000000100111010
1000000000010010
1000000000001010
1000000001000001
1000000010000011
0100010010000011
0000000100111011
1000000000010001
0010000010010010
0000000100111011
1000000000010010
1000000000001010
1000000001000001
1000000010000011
0100010011001011
0000000100111110
```

Codes for find biggest number by compare sub of new data with {1'b1,15'b0} and save bigger in Register[4] and save in memory

Biggest data will be saved in Register[4]

Memory data

```
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000100010101001
0000001010100010
0000000000011100
0000000111100000
0000000001001000
0000000100111000
0000010101110000
0011111000101000
0000001110101000
0100101010101000
0001010010100000
0010101010101000
0111001110101010
0000000101001100
0000100110110000
0000000100000101
0000000001010110
0000000001111100
0001111110101000
0000000100010100
0000010101111000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
```

Biggest data in memory

```
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0111001110101010
0000000000000000
0000000000000000
0000000001100000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
```