Very Simple (VS) MIPS Implementation

Goal is to share as much hardware as possible.

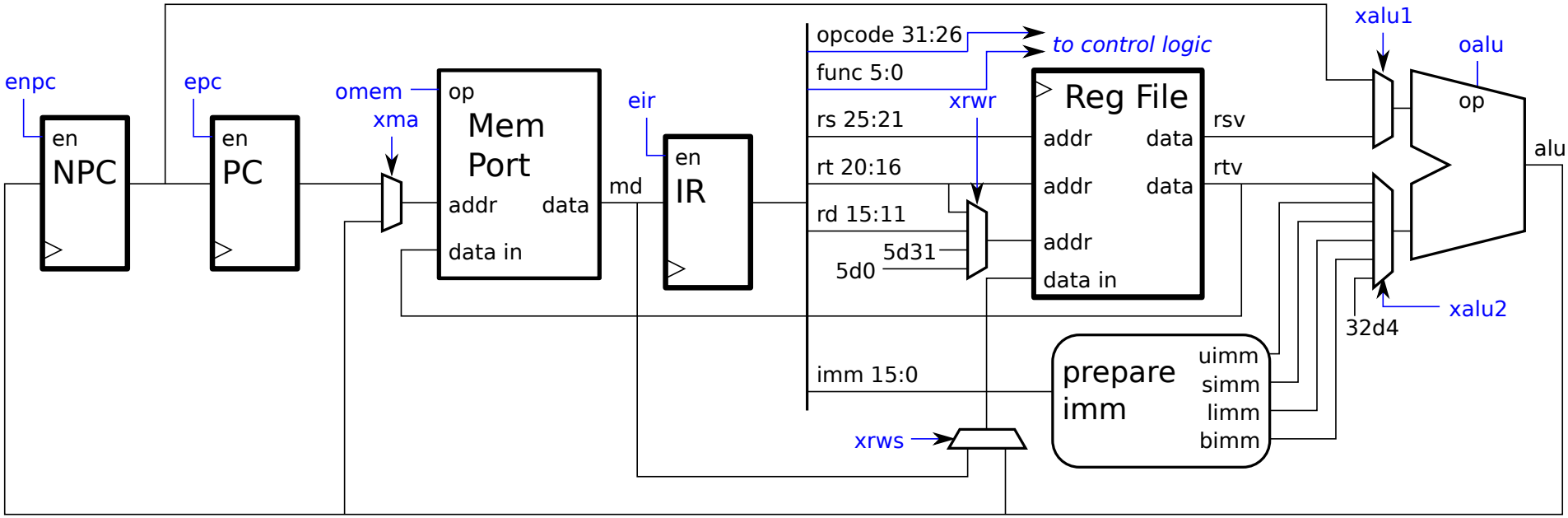
Goal is to be simple.

Good for illustrating datapath and control hardware.

With relatively minor changes VS MIPS can be made into a practical design.

mips-hw-2

Very Simple Implementation



Datapath components in black.

Control signals in blue.

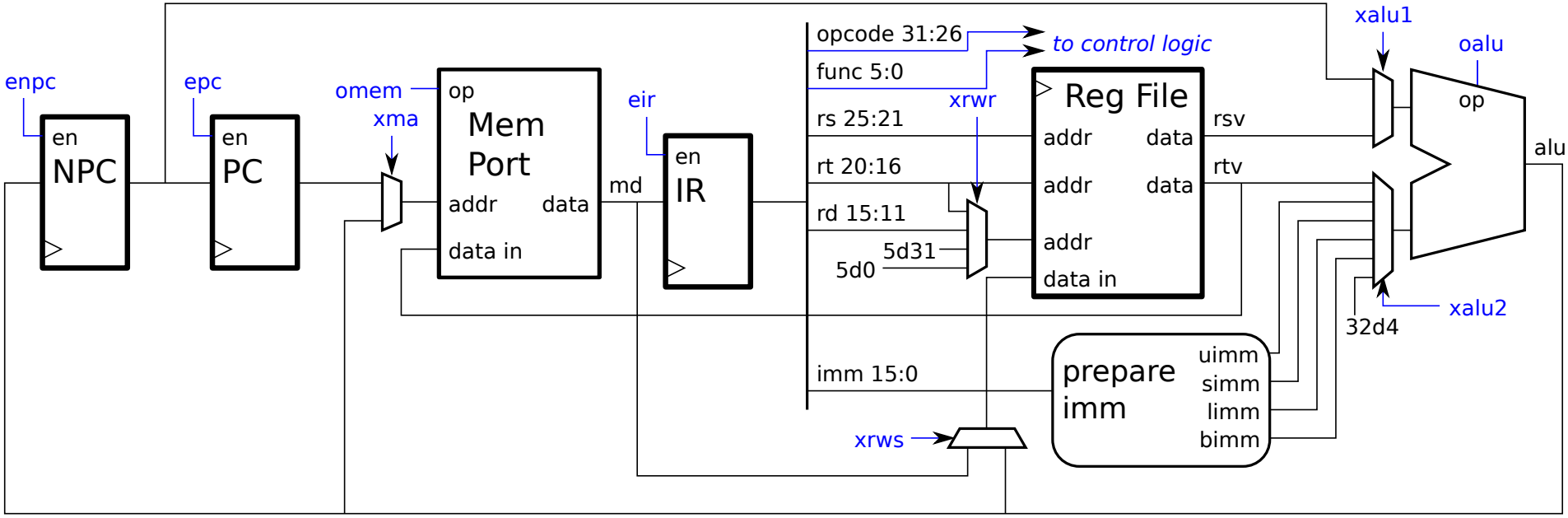
Control signals set by control logic.

Warning: This implementation impractically slow.

mips-hw-2

mips-hw-3

Control for add Instruction

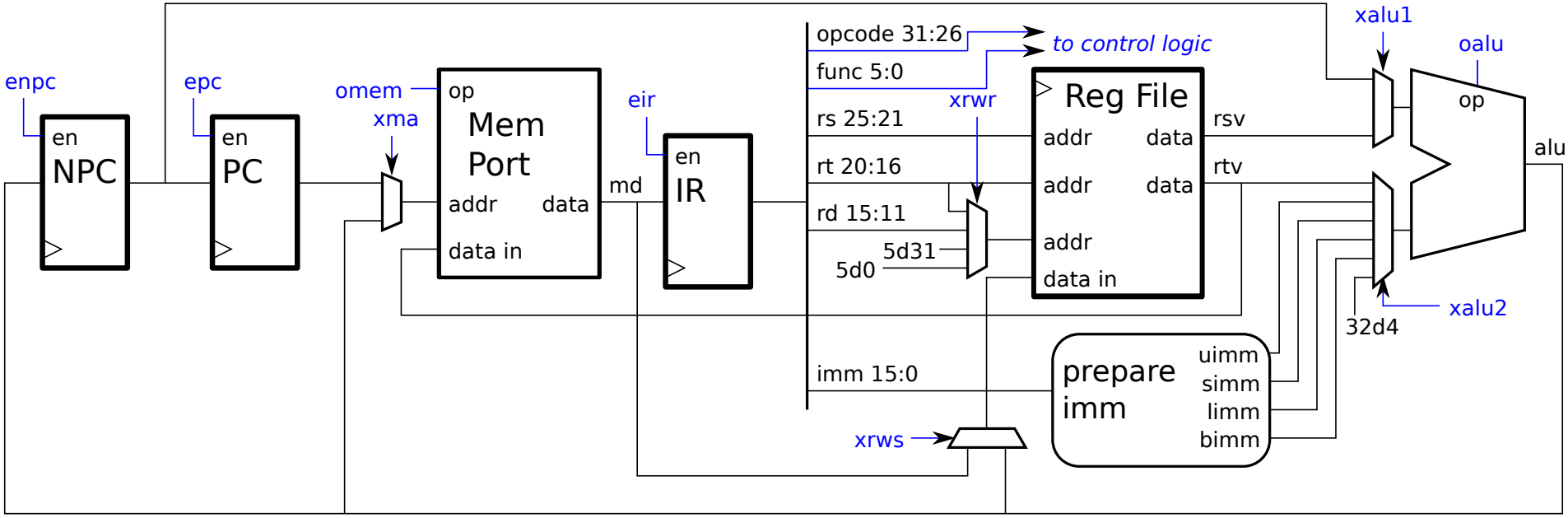


State	SubSt	xma	omem	eir	xalu1	xalu2	oalu	xrwr	xrws	enpc	epc	Next
IF		pc	rd32	1								ID
ID	add				rsv	rtv	add	rd	alu			NI
NI					npc	4	add			1	1	IF

mips-hw-3

mips-hw-4

Control for sub Instruction

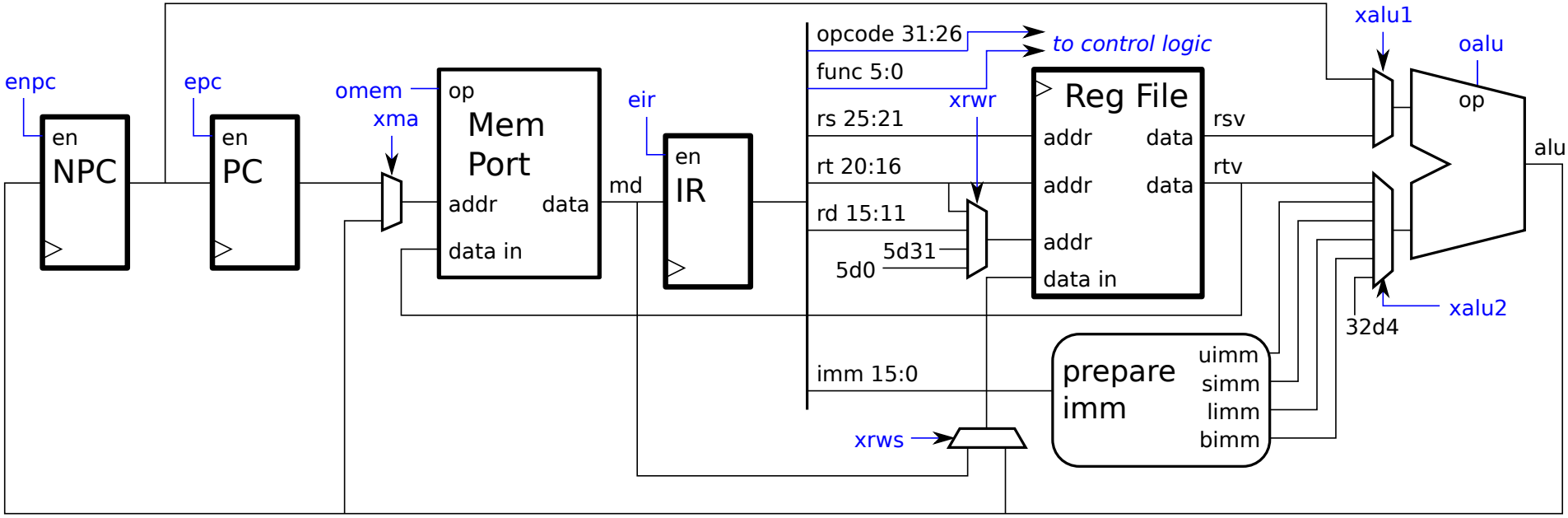


State	SubSt	xma	omem	eir	xalu1	xalu2	oalu	xrwr	xrws	enpc	epc	Next
IF		pc	rd32	1								ID
ID	sub				rsv	rtv	sub	rd	alu			NI
NI					npc	4	add			1	1	IF

mips-hw-4

mips-hw-5

Control for addi Instruction

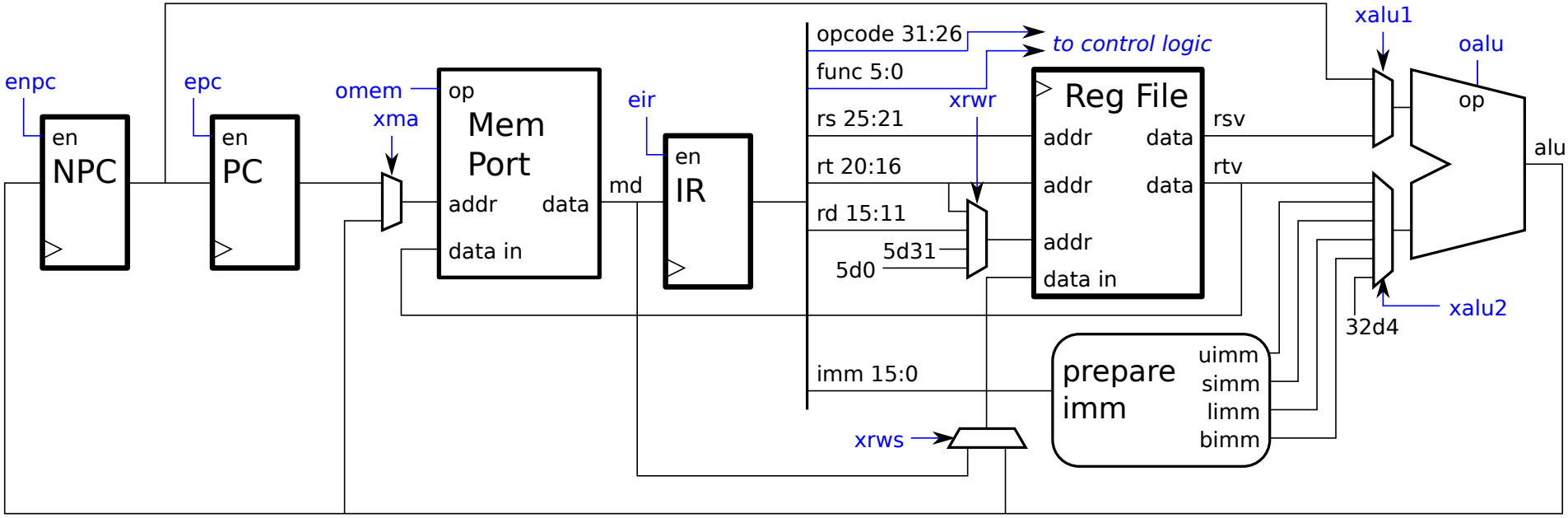


State	SubSt	xma	omem	eir	xalu1	xalu2	oalu	xrwr	xrws	enpc	epc	Next
IF		pc	rd32	1								ID
ID	addi				rsv	simm	add	rt	alu			NI
NI					npc	4	add			1	1	IF

mips-hw-5

mips-hw-6

Control for `lw` Instruction



State	SubSt	xma	omem	eir	xalu1	xalu2	oalu	xrwr	xrws	enpc	epc	Next
IF		pc	rd32	1								ID
ID	lw	alu	rd32		rsv	simm	add	rt	md			NI
NI					npc	4	add			1	1	IF

The `lw` instruction reveals a weakness in this design...

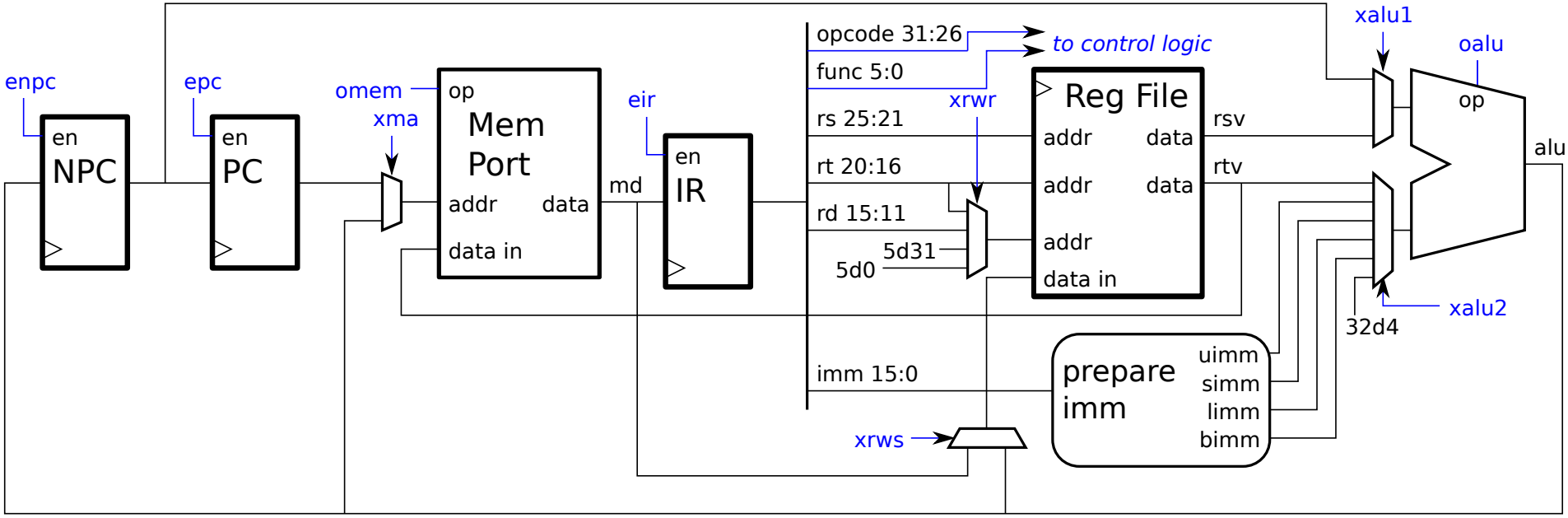
... `ID` takes much longer for `lw` than for `add` and other instructions...

... forcing the clock frequency to be slower than it has to be. (Nevermind for now.)

mips-hw-6

mips-hw-7

Control for `lb` Instruction



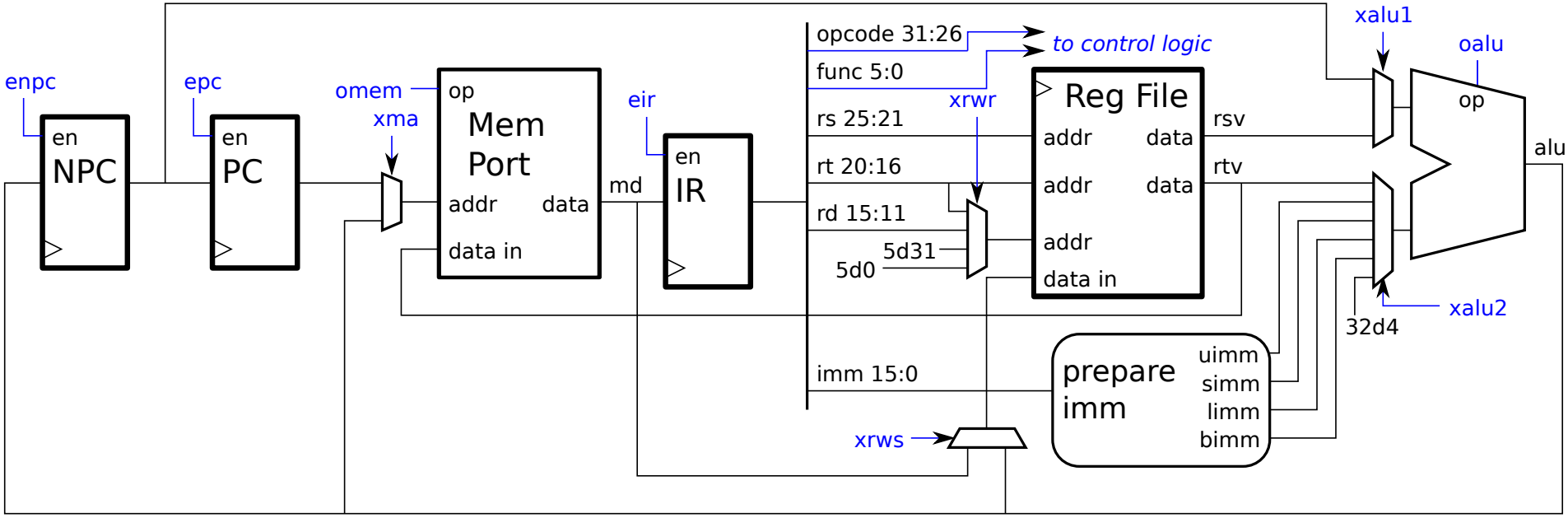
State	SubSt	xma	omem	eir	xalu1	xalu2	oalu	xrwr	xrws	enpc	epc	Next
IF		pc	rd32	1								ID
ID	lb	alu	rd8s		rsv	simm	add	rt	md			NI
NI					npc	4	add			1	1	IF

The work of sign-extending or padding the loaded value is done by the memory port.

mips-hw-7

mips-hw-8

Control for `sb` Instruction

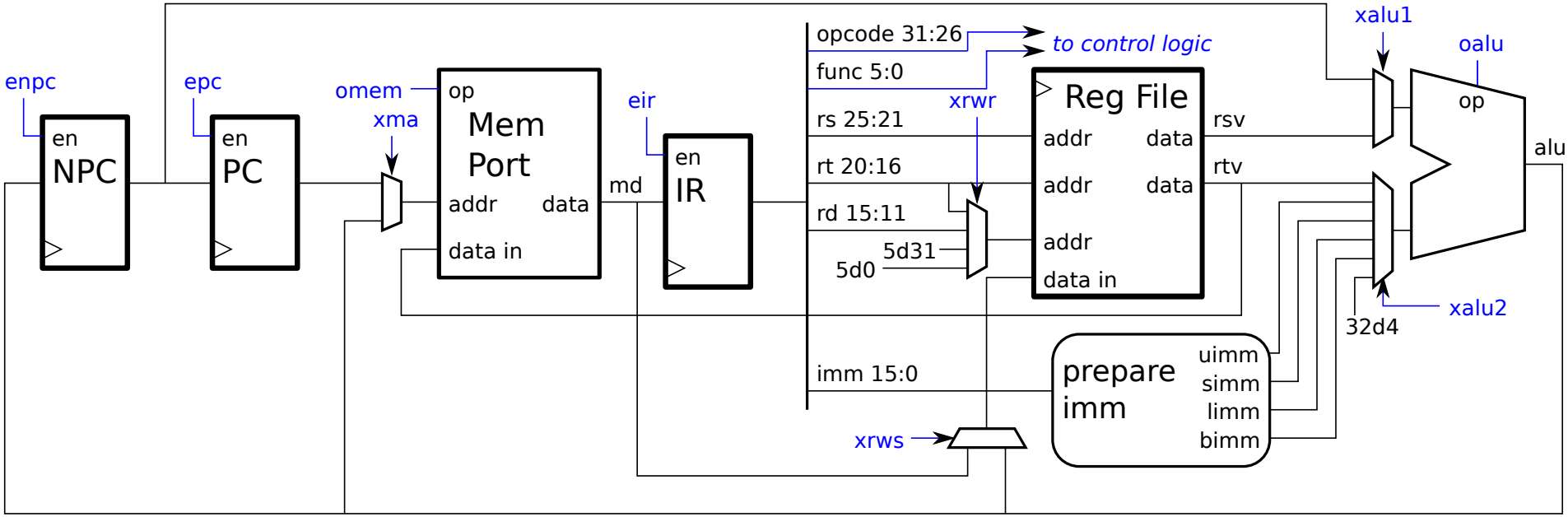


State	SubSt	xma	omem	eir	xalu1	xalu2	oalu	xrwr	xrws	enpc	epc	Next
IF		pc	rd32	1								ID
ID	sb	alu	wr8s		rsv	simm	add					NI
NI					npc	4	add			1	1	IF

mips-hw-8

mips-hw-9

Control for beq Instruction



State	SubSt	xma	omem	eir	xalu1	xalu2	oalu	xrwr	xrws	enpc	epc	Next
IF		pc	rd32	1								ID
ID	beq				rsv	rtv	seq					NI or BT
BT	beq				npc	bimm	add			1	1	IF
NI					npc	4	add			1	1	IF

mips-hw-9

Control Logic Design

Lets collect the control signals we've decided so far:

State	SubSt	xma	omem	eir	xalu1	xalu2	oalu	xrwr	xrws	enpc	epc	Next
IF		pc	rd32	1								ID
BT					npc	bimm	add			1	1	IF
NI					npc	4	add			1	1	IF
ID	add				rsv	rtv	add	rd	alu			NI
ID	sub				rsv	rtv	sub	rd	alu			NI
ID	addi				rsv	simm	add	rt	alu			NI
ID	lw	alu	rd32		rsv	simm	add	rt	md			NI
ID	lb	alu	rd8s		rsv	simm	add	rt	md			NI
ID	sb	alu	wr8s		rsv	simm	add					NI
ID	beq				rsv	rtv	seq					NI

Notice that IF, BT, and NI states were all identical ...

... but ID varies depending on the instruction.

Control Logic Design

The State Register

We'll need a register to hold the current state.

Since there are four states we can use a two-bit register.

Possible state encoding: `IF` = 0, `ID` = 1, ...

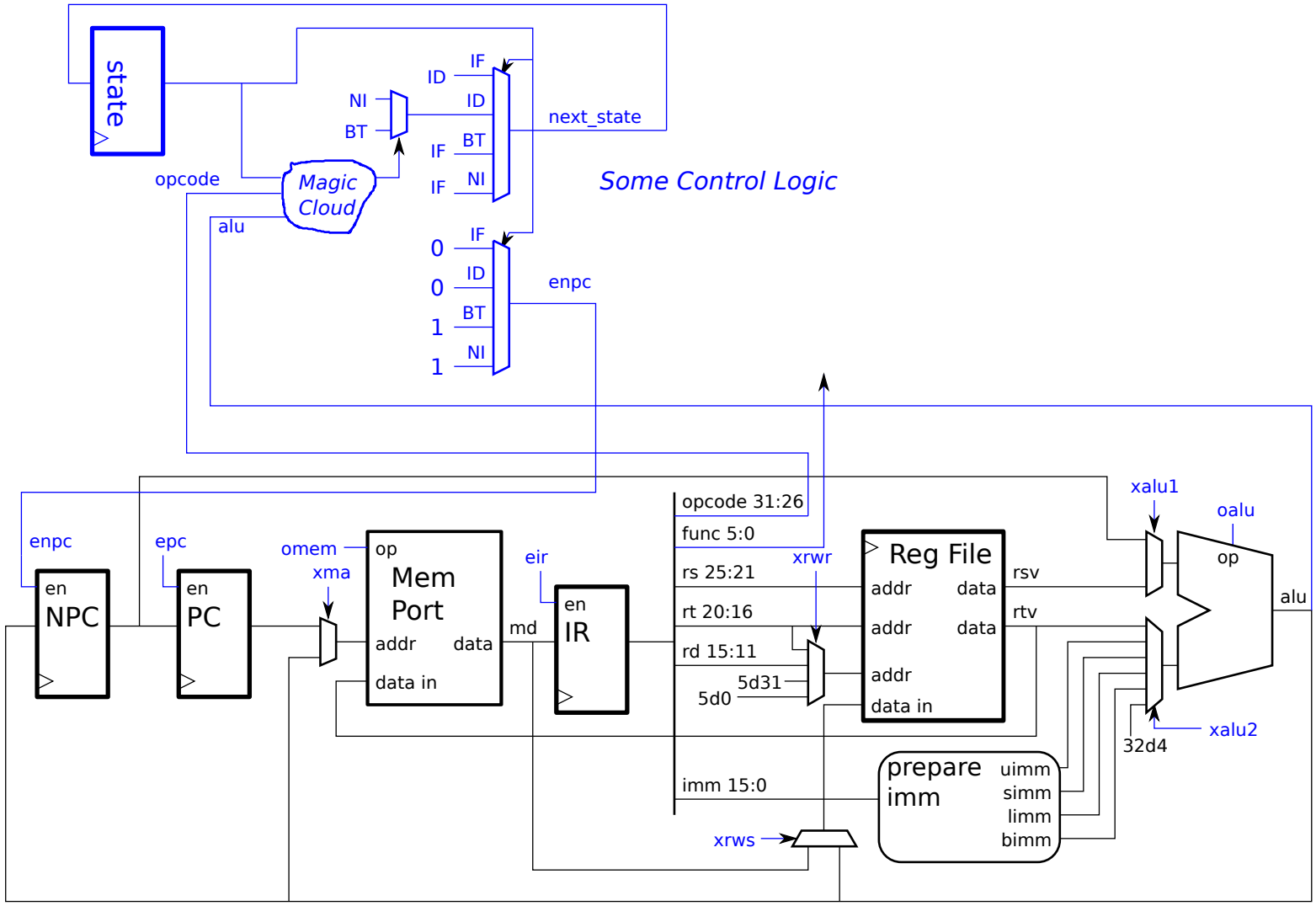
In diagrams we will use names, such as `IF`, instead of encoded values, such as `0`.

Next-State Logic

There will be combinational logic to determine the next state.

Next state is computed in terms of. . .
. . . *current state*, *opcode*, and for branches the *ALU output*.

Control Logic for State



Note: Magic Cloud checks for a taken branch.

Logic for the enpc signal.

State	SubSt	xma	omem	eir	xalu1	xalu2	oalu	xrwr	xrws	enpc	epc	Next
IF		pc	rd32	1								ID
BT					npc	bimm	add			1	1	IF
NI					npc	4	add			1	1	IF
ID	add				rsv	rtv	add	rd	alu			NI
ID	sub				rsv	rtv	sub	rd	alu			NI
ID	addi				rsv	simm	add	rt	alu			NI
ID	lw	alu	rd32		rsv	simm	add	rt	md			NI
ID	lb	alu	rd8s		rsv	simm	add	rt	md			NI
ID	sb	alu	wr8s		rsv	simm	add					NI
ID	beq				rsv	rtv	seq					NI

Notice that enpc is 1 in BT and NI and 0 in the other states.

So we need logic with an output of 1 for those two states.

An easy way to do that is a mux, we'll rely on the synthesis program to streamline the logic.

Some Control Logic