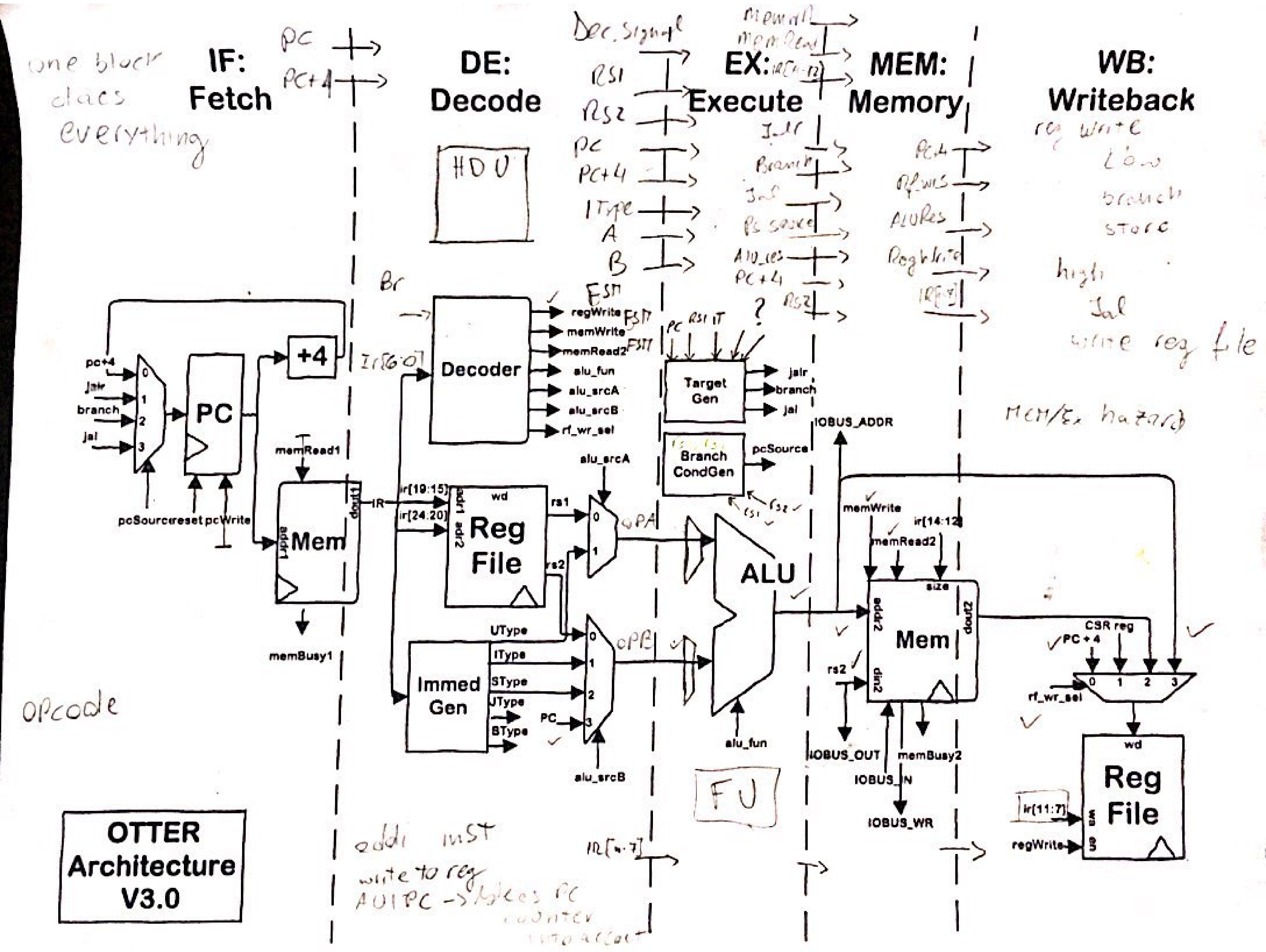
IF: Fetch
DE: Decode
EX: Execute
MEM: Memory
WB: Writeback
PC
PC+4
Dec. Signal
RS1
Rs2
PC
PC+4
IType
A
B
ESM
memwrite
memRead
IR[14:12]
Jalr
Branch
Jal
Pc source
ALU_res
PC+4
RS2
?
PC+4
Rf_wrs
ALURes
RegWrite
IR[11:7]
IR[11:7]
HDU
Br
Ir[6:0]
pc+4
jalr
branch
jal
0
1
2
3
PC
+4
pcSource
reset
pcWrite
memRead1
Mem
addr1
dout1
memBusy1
IR
Decoder
regWrite
memWrite
memRead2
alu_fun
alu_srcA
alu_srcB
rf_wr_sel
ESM
ESM
ir[19:15]
ir[24:20]
adr1
adr2
wd
Reg File
rs1
rs2
alu_srcA
0
1
opA
UType
IType
SType
JType
BType
Immed Gen
PC
0
1
2
3
opB
alu_srcB
PC
RS1
IT
Target Gen
jalr
branch
jal
rs1
rs2
Branch CondGen
pcSource
ALU
alu_fun
FU
IOBUS_ADDR
memWrite
memRead2
ir[14:12]
size
addr2
din2
dout2
Mem
rs2
IOBUS_OUT
memBusy2
IOBUS_IN
IOBUS_WR
CSR reg
PC + 4
0
1
2
3
rf_wr_sel
wd
Reg File
ir[11:7]
wa
regWrite
en
OTTER Architecture V3.0

one block does everything

opcode

addi inst
write to reg
AUIPC -> takes PC counter into account

reg write
low
branch
store
high
Jal
write reg file

MEM/Ex hazard

Scanned with CamScanner