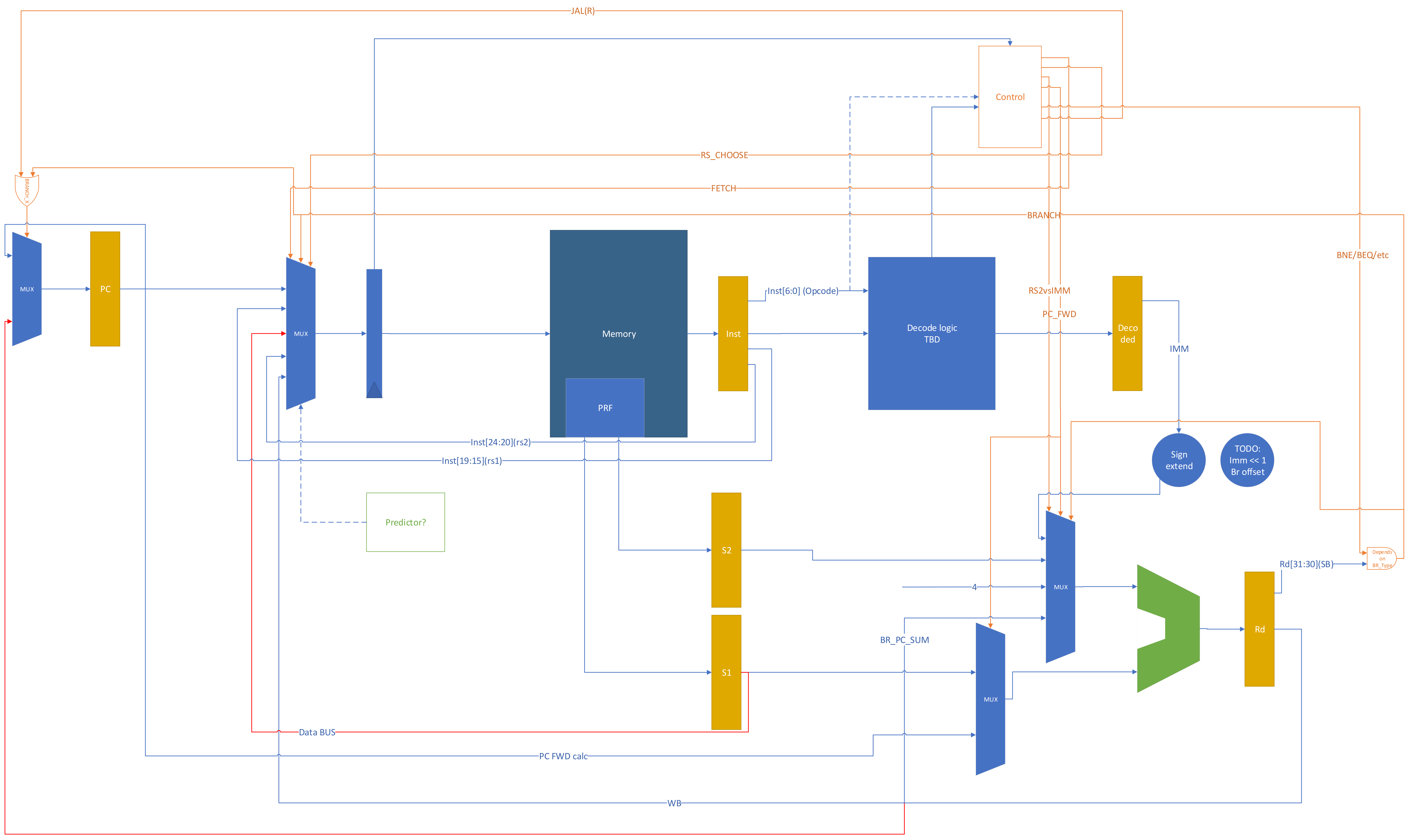
opcode	imm	funct3	funct7
rs1	rs2	rd	

instruction	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Comment
R-type																										opcode							
I-type	s																									opcode							
S-type	s																									opcode							
U-type	s																									opcode							
B-type (S)	im12	imm[10:5]																			imm[4:1]				im11	opcode							Same as S but imm*2
J-type (U)	im20	imm[10:1]										im11	imm[19:12]													opcode							imm shifted by 12 bits
nop	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ADDI			0	0	0	0	0	OP-IMM							
Arithmetics																					dest					OP							
ADD/SLT/SLTU	0	0	0	0	0	0	0	src2					src1					ADD/SLT/SLTU			dest					OP							
AND/OR/XOR	0	0	0	0	0	0	0	src2					src1					AND/OR/XOR			dest					OP							
SLL/SRL	0	0	0	0	0	0	0	src2					src1					SLL/SRL			dest					OP							
SUB/SRA	0	1	0	0	0	0	0	src2					src1					SUB/SRA			dest					OP							
Control flow																																	
JAL	im20	imm[10:1]										im11	imm[19:12]								dest					JAL							plain unconditional jump
JALR	offset[11:0]												base					0	0	0	dest					JALR							
BEQ/BNE	offset[12|10:5]							src2					src1					BEQ/BNE			offset[11|4:1]					BRANCH							
BLT(U)	offset[12|10:5]							src2					src1					BLT(U)			offset[11|4:1]					BRANCH							
BGE(U)	offset[12|10:5]							src2					src1					BGE(U)			offset[11|4:1]					BRANCH							branch if rs1 > rs2 (U - unsigned)
Memory																																	
Load	offset[11:0]												base					width			dest					LOAD							Byte address = rs1 + sign_ext 12-bit offset. copy memo
Store	offset[11:5]							src					base					width			offset[4:0]					STORE							copy value rs2 into memory
Fence	fm?	0	0	0	PI	PO	PR	PW	SI	SO	SR	SW	0	0	0	0	0	fence			0	0	0	0	0	MISC-MEM							fm=0000 for normal fence, 1000 for (TSO) exclude write
System																																	
ECALL	ecall												0	0	0	0	0	PRIV			0	0	0	0	0	SYSTEM							service req to exe env
EBREAK	ebreak												0	0	0	0	0	PRIV			0	0	0	0	0	SYSTEM							return control to a debugging environment
HINT																																	
TODO!																																	
M-extension																					dest					OP							

	Instructions flow unoptimized (mem op in yellow, alu in red)					
cycle	R-TYPE	Load (I-type)	Store (S-type)	Branch (JALR)	Branch (JAL)	Branch (BNE etc)
1	Fetch	Fetch	Fetch	Fetch	Fetch	Fetch
2	Decode	Decode	Decode	Decode	Decode	Decode
3	Read src1	Read src1(base)	Read src1(base)	Read src1	PC+4 (Call->rd)	Read src1
4	Read src2	Exe (src1+imm_SE)	Read src2	PC+4 (Call->rd)	PC+imm_SE	Read src2
5	Exe	Mem (load)	Exe	Exe(src1+imm_SE<<1)	WB(rd=x1/x5)	PC+4 (Call->rd)
6	WB	WB	WB	WB(rd=x1/x5)		Compare
7	PC+4	PC+4	PC+4			PC+imm_SE

	Instructions flow optimized (mem op in yellow, alu in red)					
cycle	R-TYPE	Load (I-type)	Store (S-type)	Branch (JALR)	Branch (JAL)	Branch (BNE etc)
1	D+RS1	D+RS1	D+RS1	D+RS1	D+RS1	D+RS1
2	Read src2	Exe (src1+imm_SE) + F	Read src2	Exe(src1+imm_SE<<1)	PC+imm_SE	Read src2
3	Exe + Fetch	Mem (load)	Exe + Fetch	Fetch	Fetch	Compare
4	WB + PC	WB + PC	WB + PC	WB(rd=x1/x5)+PC	WB(rd=x1/x5)+PC	PC+imm_SE/4
5						Fetch

cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
inst	add				load				branch_taken				cond branch_taken					sub			
Fetched	D+RS1	RS2	Exe&F	WB+PC	D+RS1	Exe&F	Mem	WB+PC	D+RS1	Exe	Fetch	WB+PC	D+RS1	RS2	Compare	Exe	Fetch				
PC	4_ld	4_ld	**4_ld**	4+4=8_br	8_br	**8_br**	8_br	+8=12_??	12_??	12_??	**40_brc**	+4=44_??	+4=44_??	+4=44_??	+4=44_??	+4=44_??	**80_sub**				
R1	r1_add_1	r1_add_1	**r1_add_1**	r1_add_1	r1_ld	r1_ld	r1_ld	r1_ld	r1_br	r1_br	r1_br	r1_br	r1_brc	r1_brc	**r1_brc**	r1_brc	r1_brc				
R2		r2_add_1	**r2_add_1**	r2_add_1	r2_add_1	r2_add_1	r2_add_1	r2_add_1	r2_add_1	r2_add_1	r2_add_1	r2_add_1	r2_add_1	r2_brc	**r2_brc**	r2_brc	r2_brc				
Rd			rd=add_1	**add_1**	add_1	rd=ld_1	**ld_1**	ld_1	ld_1	rd=r1+imm	**br_1**	**br_1**	br_1	br_1	comp	**comp**	comp				
Decoded																					
Inst	add	add	**load**	load	load	**branch**	branch	branch	branch	branch	**cond_br**	cond_br	cond_br	cond_br	cond_br	cond_br	**sub**	sub			



TBD: possible enhancement
with a predictor

	Instructions flow optimized (mem op in yellow, alu in red), w/ predictor					
cycle	R-TYPE	Load (I-type)	Store (S-type)	Branch (JALR)	Branch (JAL)	Branch (BNE etc)
1	D+RS1	D+RS1	D+RS1	D+RS1 & pred_check	D+RS1 & pred_check	D+RS1 & pred_check
2	Read src2	Exe (src1+imm_SE) + F	Read src2	Exe & Fetch	PC+imm_SE & Fetch	Read src2 & PC+imm/4
3	Exe + Fetch	Mem (load)	Exe + Fetch	WB(rd=x1/x5)+PC	WB(rd=x1/x5)+PC	Compare + Fetch
4	WB + PC	WB + PC	WB + PC			