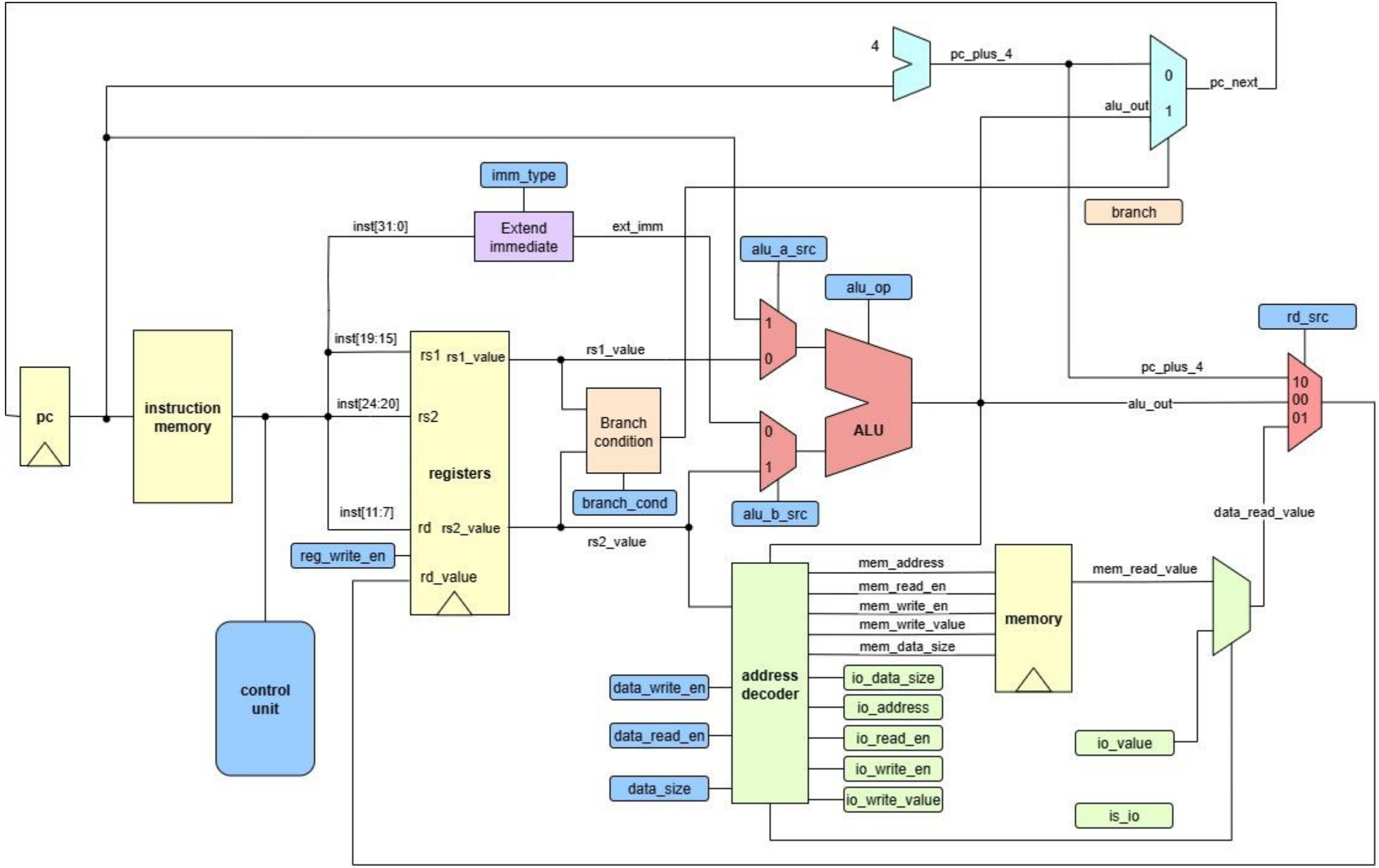
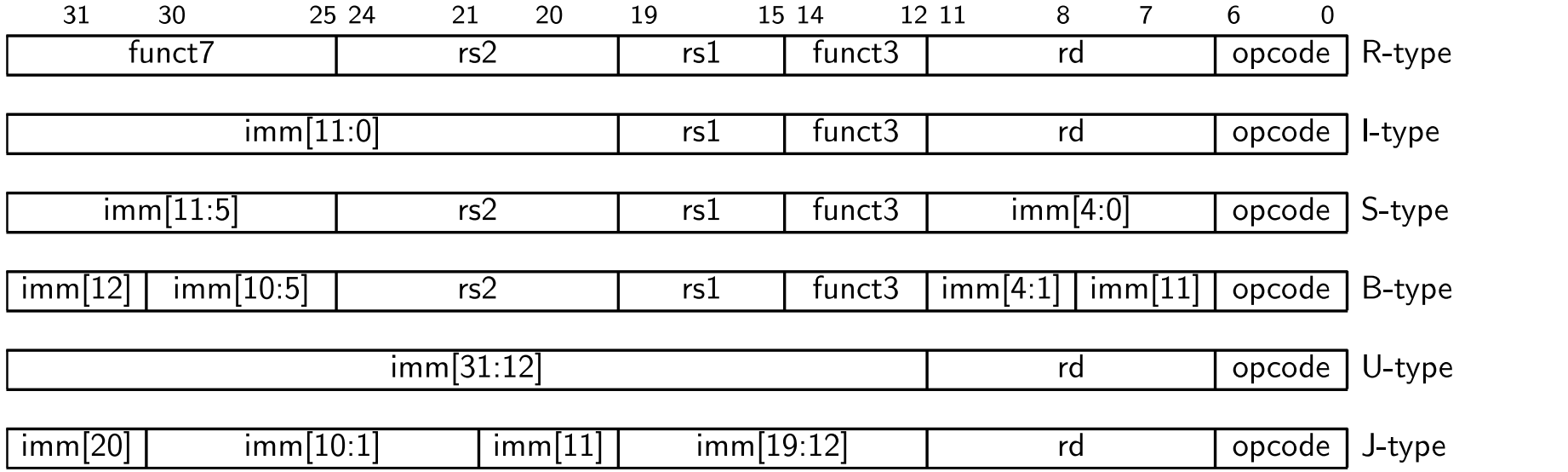
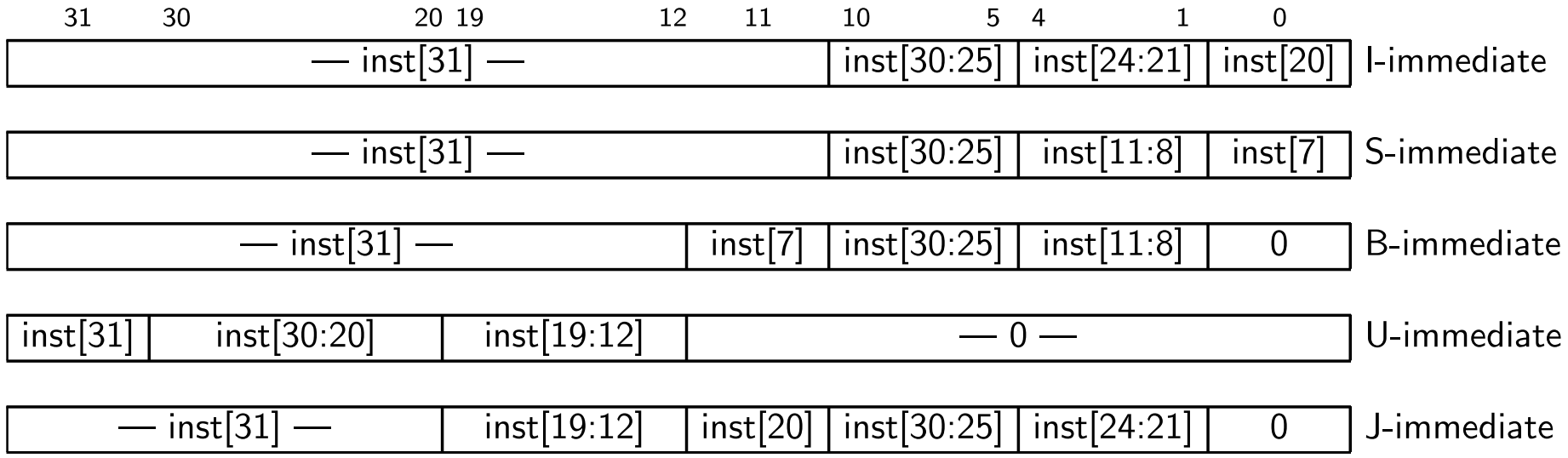
RISC V


4
pc_plus_4
0
1
pc_next
alu_out
branch
imm_type
Extend immediate
inst[31:0]
ext_imm
alu_a_src
alu_op
rd_src
pc
instruction memory
inst[19:15]
inst[24:20]
inst[11:7]
rs1 rs1_value
rs2
registers
rd rs2_value
rd_value
rs1_value
Branch condition
branch_cond
rs2_value
1
0
0
1
ALU
alu_b_src
pc_plus_4
10
00
01
alu_out
data_read_value
reg_write_en
control unit
mem_address
mem_read_en
mem_write_en
mem_write_value
mem_data_size
memory
mem_read_value
address decoder
data_write_en
data_read_en
data_size
io_data_size
io_address
io_read_en
io_write_en
io_write_value
io_value
is_io

		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	
Load	I	imm[11:0]												rs1					funct3			rd					op								12
Store	S	imm[11:5]							rs2					rs1					funct3			imm[4:0]					op								12
Data	R	funct7							rs2					rs1					funct3			rd					op								
Load imm	U	imm[31:12]																				rd					op								20
Branch	B	i[12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				i[11]	op								12
Jump	J	i[20]	imm[10:1]										i[11]	imm[19:12]								rd					op								20

		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
Load	I	inst[31]																					inst[30:25]						inst[24:21]				i[20]
Store	S	inst[31]																					inst[30:25]						inst[11:8]				i[7]
Load imm	U	i[31]	inst[30:20]											inst[19:12]								0												
Branch	B	inst[31]																				i[7]	inst[30:25]						inst[11:8]				0
Jump	J	inst[31]												imm[19:12]								i[20]	inst[30:25]						inst[24:21]				0


31 30 25 24 21 20 19 15 14 12 11 8 7 6 0
funct7 rs2 rs1 funct3 rd opcode R-type
imm[11:0] rs1 funct3 rd opcode I-type
imm[11:5] rs2 rs1 funct3 imm[4:0] opcode S-type
imm[12] imm[10:5] rs2 rs1 funct3 imm[4:1] imm[11] opcode B-type
imm[31:12] rd opcode U-type
imm[20] imm[10:1] imm[11] imm[19:12] rd opcode J-type


31 30 20 19 12 11 10 5 4 1 0
— inst[31] — inst[30:25] inst[24:21] inst[20] I-immediate
— inst[31] — inst[30:25] inst[11:8] inst[7] S-immediate
— inst[31] — inst[7] inst[30:25] inst[11:8] 0 B-immediate
inst[31] inst[30:20] inst[19:12] — 0 — U-immediate
— inst[31] — inst[19:12] inst[20] inst[30:25] inst[24:21] 0 J-immediate

RISC-V RV32IM ISA Reference Sheet v1.4

31 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]		rs1	funct3	rd	opcode	I-type
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12,10:5]	rs2	rs1	funct3	imm[4:1,11]	opcode	B-type
imm[31:12]				rd	opcode	U-type
imm[20,10:1,11,19:12]				rd	opcode	J-type

instruction		fmt	opcode	fun3	fun7	semantics	encoding
lui	rd,imm20	U	0x37			rd = imm20 << 12	iiii iiii iiii iiii iiii dddd d011 0111
auipc	rd,imm20	U	0x17			rd = pc + (imm20 << 12)	iiii iiii iiii iiii iiii dddd d001 0111
addi	rd,rs1,imm12	I	0x13	000		rd = rs1 + se(imm12)	iiii iiii iiii ssss s000 dddd d001 0011
slti	rd,rs1,imm12	I	0x13	010		rd = rs1 <signed se(imm12) ? 1 : 0	iiii iiii iiii ssss s010 dddd d001 0011
sltiu	rd,rs1,imm12	I	0x13	011		rd = rs1 <unsign se(imm12) ? 1 : 0	iiii iiii iiii ssss s011 dddd d001 0011
xori	rd,rs1,imm12	I	0x13	100		rd = rs1 ^ se(imm12)	iiii iiii iiii ssss s100 dddd d001 0011
ori	rd,rs1,imm12	I	0x13	110		rd = rs1 | se(imm12)	iiii iiii iiii ssss s110 dddd d001 0011
andi	rd,rs1,imm12	I	0x13	111		rd = rs1 & se(imm12)	iiii iiii iiii ssss s111 dddd d001 0011
slli	rd,rs1,imm12	I	0x13	001	0x0	rd = rs1 << imm12[4:0]	0000 000i iiii ssss s001 dddd d001 0011
srli	rd,rs1,imm12	I	0x13	101	0x0	rd = rs1 >> imm12[4:0]	0000 000i iiii ssss s101 dddd d001 0011
srai	rd,rs1,imm12	I	0x13	101	0x20	rd = rs1 >>> imm12[4:0]	0100 000i iiii ssss s101 dddd d001 0011
add	rd,rs1,rs2	R	0x33	000	0x0	rd = rs1 + rs2	0000 000t tttt ssss s000 dddd d011 0011
sub	rd,rs1,rs2	R	0x33	000	0x20	rd = rs1 - rs2	0100 000t tttt ssss s000 dddd d011 0011
sll	rd,rs1,rs2	R	0x33	001	0x0	rd = rs1 << rs2[4:0]	0000 000t tttt ssss s001 dddd d011 0011
slt	rd,rs1,rs2	R	0x33	010	0x0	rd = rs1 <signed rs2 ? 1 : 0	0000 000t tttt ssss s010 dddd d011 0011
sltu	rd,rs1,rs2	R	0x33	011	0x0	rd = rs1 <unsign rs2 ? 1 : 0	0000 000t tttt ssss s011 dddd d011 0011
xor	rd,rs1,rs2	R	0x33	100	0x0	rd = rs1 ^ rs2	0000 000t tttt ssss s100 dddd d011 0011
srl	rd,rs1,rs2	R	0x33	101	0x0	rd = rs1 >> rs2[4:0]	0000 000t tttt ssss s101 dddd d011 0011
sra	rd,rs1,rs2	R	0x33	101	0x20	rd = rs1 >>> rs2[4:0]	0100 000t tttt ssss s101 dddd d011 0011
or	rd,rs1,rs2	R	0x33	110	0x0	rd = rs1 | rs2	0000 000t tttt ssss s110 dddd d011 0011
and	rd,rs1,rs2	R	0x33	111	0x0	rd = rs1 & rs2	0000 000t tttt ssss s111 dddd d011 0011
lb	rd,imm12(rs1)	I	0x03	000		rd = se(mem[rs1+se(imm12)][7:0])	iiii iiii iiii ssss s000 dddd d000 0011
lh	rd,imm12(rs1)	I	0x03	001		rd = se(mem[rs1+se(imm12)][15:0])	iiii iiii iiii ssss s001 dddd d000 0011
lw	rd,imm12(rs1)	I	0x03	010		rd = mem[rs1+se(imm12)][31:0]	iiii iiii iiii ssss s010 dddd d000 0011
lbu	rd,imm12(rs1)	I	0x03	100		rd = ze(mem[rs1+se(imm12)][7:0])	iiii iiii iiii ssss s100 dddd d000 0011
lhu	rd,imm12(rs1)	I	0x03	101		rd = ze(mem[rs1+se(imm12)][15:0])	iiii iiii iiii ssss s101 dddd d000 0011
sb	rs2,imm12(rs1)	S	0x23	000		mem[rs1+se(imm12)][7:0] = rs2[7:0]	iiii iiit tttt ssss s000 iiii i010 0011
sh	rs2,imm12(rs1)	S	0x23	001		mem[rs1+se(imm12)][15:0] = rs2[15:0]	iiii iiit tttt ssss s001 iiii i010 0011
sw	rs2,imm12(rs1)	S	0x23	010		mem[rs1+se(imm12)][31:0] = rs2	iiii iiit tttt ssss s010 iiii i010 0011
jal	rd,targ20	J	0x6f			rd = pc+4; pc += se(targ20<<1)	iiii iiii iiii iiii iiii dddd d110 1111
jalr	rd,imm12(rs1)	I	0x67	000		rd = pc+4; pc = (rs1+se(imm12)) & ~0x1	iiii iiii iiii ssss s000 dddd d110 0111

beq	rs1,rs2,targ12	B	0x63	000		if (rs1 == rs2) pc += se(targ12<<1)	iiii iiit tttt ssss s000 iiii i110 0011
bne	rs1,rs2,targ12	B	0x63	001		if (rs1 != rs2) pc += se(targ12<<1)	iiii iiit tttt ssss s001 iiii i110 0011
blt	rs1,rs2,targ12	B	0x63	100		if (rs1 <signed rs2) pc += se(targ12<<1)	iiii iiit tttt ssss s100 iiii i110 0011
bge	rs1,rs2,targ12	B	0x63	101		if (rs1 ≥signed rs2) pc += se(targ12<<1)	iiii iiit tttt ssss s101 iiii i110 0011
bltu	rs1,rs2,targ12	B	0x63	110		if (rs1 <unsign rs2) pc += se(targ12<<1)	iiii iiit tttt ssss s110 iiii i110 0011
bgeu	rs1,rs2,targ12	B	0x63	111		if (rs1 ≥unsign rs2) pc += se(targ12<<1)	iiii iiit tttt ssss s111 iiii i110 0011
mul	rd,rs1,rs2	R	0x33	000	0x01	rd = (rs1 * rs2)[31:0]	0000 001t tttt ssss s000 dddd d011 0011
mulh	rd,rs1,rs2	R	0x33	001	0x01	rd = (signed(rs1) * signed(rs2))[63:32]	0000 001t tttt ssss s001 dddd d011 0011
mulhsu	rd,rs1,rs2	R	0x33	010	0x01	rd = (signed(rs1) * unsign(rs2))[63:32]	0000 001t tttt ssss s010 dddd d011 0011
mulhu	rd,rs1,rs2	R	0x33	011	0x01	rd = (unsign(rs1) * unsign(rs2))[63:32]	0000 001t tttt ssss s011 dddd d011 0011
div	rd,rs1,rs2	R	0x33	100	0x01	rd = rs1 /signed rs2	0000 001t tttt ssss s100 dddd d011 0011
divu	rd,rs1,rs2	R	0x33	101	0x01	rd = rs1 /unsign rs2	0000 001t tttt ssss s101 dddd d011 0011
rem	rd,rs1,rs2	R	0x33	110	0x01	rd = rs1 %signed rs2	0000 001t tttt ssss s110 dddd d011 0011
remu	rd,rs1,rs2	R	0x33	111	0x01	rd = rs1 %unsign rs2	0000 001t tttt ssss s111 dddd d011 0011

lui, auipc, jal, jalr, b immediates

Assembler:	lui x1, 0xf_ffff	20-bit number, unsigned
Real value:	0xffff_f000	Top 20 bits
Encoded value:	0xffff_f000	
Type:	U type	

Instruction	Encoded Value	Assembler Value
00000000000000000000	0	0
00000000000000000001	1	1
11111111111111111111	1 048 575	1 408 575
10000000000000000000	524 288	524288

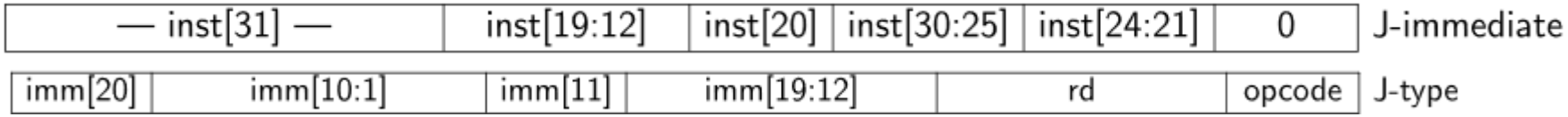
lui, auipc, jal, jalr, b immediates

Assembler:	aupic x1, 0xf_ffff	20-bit number, unsigned
Real value:	0xffff_f000	Top 20 bits
Encoded value:	0xffff_f000	
Type:	U type	

Instruction	Encoded Value	Assembler Value
00000000000000000000	0	0
00000000000000000001	1	1
11111111111111111111	1 048 575	1 408 575
10000000000000000000	524 288	524288

lui, auipc, jal, jalr,b immediates

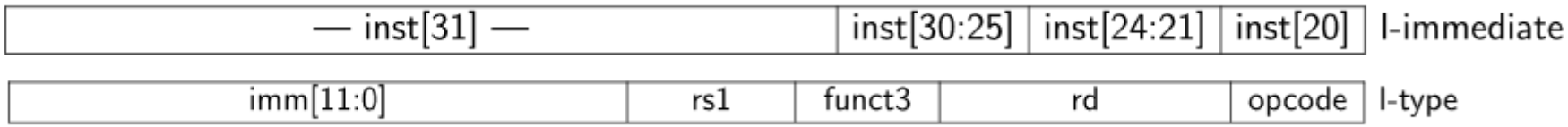
Assembler:	jal x1, 0x1f_fffe	21-bit, imm[20:1], imm[0] is 0
Real value:	0x001f_fffe	
Encoded value:	0x000f_ffff	20-bit, imm[0] is not stored
Type:	J type	
Note:	As bit 0 is 0, only 20 bits are encoded into the instruction	



Instruction	Encoded Value	Assembler Value
0_0000000000_0_00000000_00001_11011_11	0	0
0_0000000001_0_00000000_00001_11011_11	1	2
0_1111111111_0_00000000_00001_11011_11	1023	2046
0_0000000000_1_00000000_00001_11011_11	1024	2048
0_1111111111_1_00000000_00001_11011_11	2047	4094
0_1111111111_1_00000001_00001_11011_11	2048	4096
0_1111111111_1_11111111_00001_11011_11	524 287	1 048 574
1_0000000000_0_00000000_00001_11011_11	-524 288	-1 048 576
1_0000000001_0_00000000_00001_11011_11	-523 287	-1 048 574
1_1111111111_0_00000000_00001_11011_11	-523 265	-1 046 530
1_1111111111_1_00000000_00001_11011_11	-522 241	-1 044 482
1_1111111111_1_00000001_00001_11011_11	-520 192	-1 040 386
1_1111111111_1_11111111_00001_11011_11	-1	-2

lui, auipc, jal, jalr,b immediates

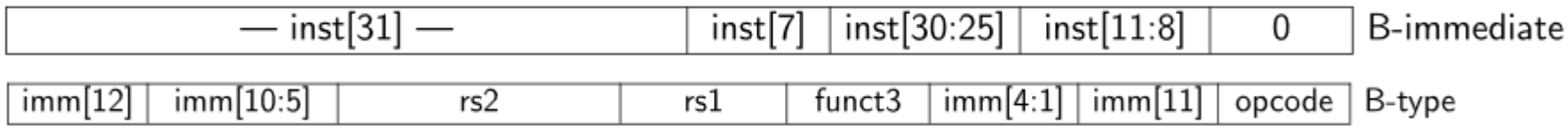
Assembler: jalr x1, 0xfff(x2) 12-bit number
Real value: 0x0000_0fff
Encoded value: 0x0000_0fff
Type: I type
Note: Bottom bit set to 0 after adding to rs1



Instruction	Encoded Value	Assembler Value
000000000000_00010_000_00001_11001_11	0	0
000000000001_00010_000_00001_11001_11	1	1
011111111111_00010_000_00001_11001_11	2047	2047
100000000000_00010_000_00001_11001_11	-2048	-2048
111111111111_00010_000_00001_11001_11	-1	-1

lui, auipc, jal, jalr,b immediates

Assembler:	beq x1, x2, 0x000_1ffe	13-bit imm[12:1], imm[0] is 0
Real value:	0x0000_1ffe	
Encoded value:	0x0000_0fff	12-bit, imm[0] is not stored
Type:	B type	
Note:	As bit 0 is 0, only 12 bits are encoded into the instruction	



Instruction	Encoded Value	Assembler Value
0_000000_00010_00001_000_**0000_0**_11000_11	0	0
0_000000_00010_00001_000_**0001_0**_11000_11	1	2
0_000000_00010_00001_000_**1111_0**_11000_11	15	30
0_000001_00010_00001_000_**0000_0**_11000_11	16	32
0_111111_00010_00001_000_**1111_0**_11000_11	1023	2046
0_000000_00010_00001_000_**0000_1**_11000_11	1024	2048
0_000000_00010_00001_000_**0001_1**_11000_11	1025	2050
0_111111_00010_00001_000_**1111_1**_11000_11	2047	4094
1_111111_00010_00001_000_**1111_1**_11000_11	-1	-2
1_000000_00010_00001_000_**0001_1**_11000_11	-1028	-2046
1_000000_00010_00001_000_**0000_0**_11000_11	-2048	-4096

Inst Type	Bits	Hex max	Total range	Unsigned max	Bit Range		Instruction Range
I	12	0x00 00 0F FF	4096	4095	-2048	2047	-2048 to 2047
S	12	0x00 00 0F FF	4096	4095	-2048	2047	-2048 to 2047
U	20	0x00 0F FF FF	1 048 576	1 048 575	-524 288 0	524 287 1 048 575	-2 147 483 648 to 2 147 483 647 in steps of 4096 0 to 4294967295 in steps of 4096
B	12	0x00 00 0F FF	4096	4095	-2048	2047	-4096 to 4095 in steps of 2
J	20	0x00 0F FF FF	1 048 576	1 048 575	-524 288	524 287	-1 048 576 to 1 048 575 in steps of 2
J	21	0x00 1F FF FF	2 097 152	2 097 151	-1 048 576	1 048 575	

Control signals											
instruction	**opcode**	**alu_a_src**	**alu_b_src**	**rd_src**	**reg_write_en**	**data_read_en**	**data_write_en**	**data_size**	**alu_op**	**imm_type**	**branch_cond**
arithmetic reg	011_0011	0	0	00	1	0	0	000	see table	0	010
arithmetic imm	001_0011	0	1	00	1	0	0	000	see table	1	010
load	000_0011	0	1	01	1	1	0	see table	0000	1	010
store	010_0011	0	1	00	0	0	1	see table	0000	2	010
branch	110_0011	1	1	00	0	0	0	000	0000	3	see table
jal	110_1111	1	1	00	0	0	0	000	0000	4	011
jalr	110_0111	0	1	10	1	0	0	000	0000	1	011
lui	011_0111	0	1	00	1	0	0	000	0000	5	010
auipc	001_0111	1	1	00	1	0	0	000	0000	5	010

Immediate encoding			
Instruction	**Opcode**	**Immediate Type**	**Immediate Code**
add	011_0011	R	0
addi	001_0011	I	1
lw	000_0011	I	1
sw	010_0011	S	2
beq	110_0011	B	3
jal	110_1111	J	4
jalr	110_0111	I	1
lui	011_0111	U	5
auipc	001_0111	U	5

Immediate types	
Immediate Type	**Immediate Code**
R	0
I	1
S	2
B	3
J	4
U	5

alu_a_src	
Value	**Source**
0	rs1
1	pc

alu_b_src	
Value	**Source**
0	rs2
1	ext_imm

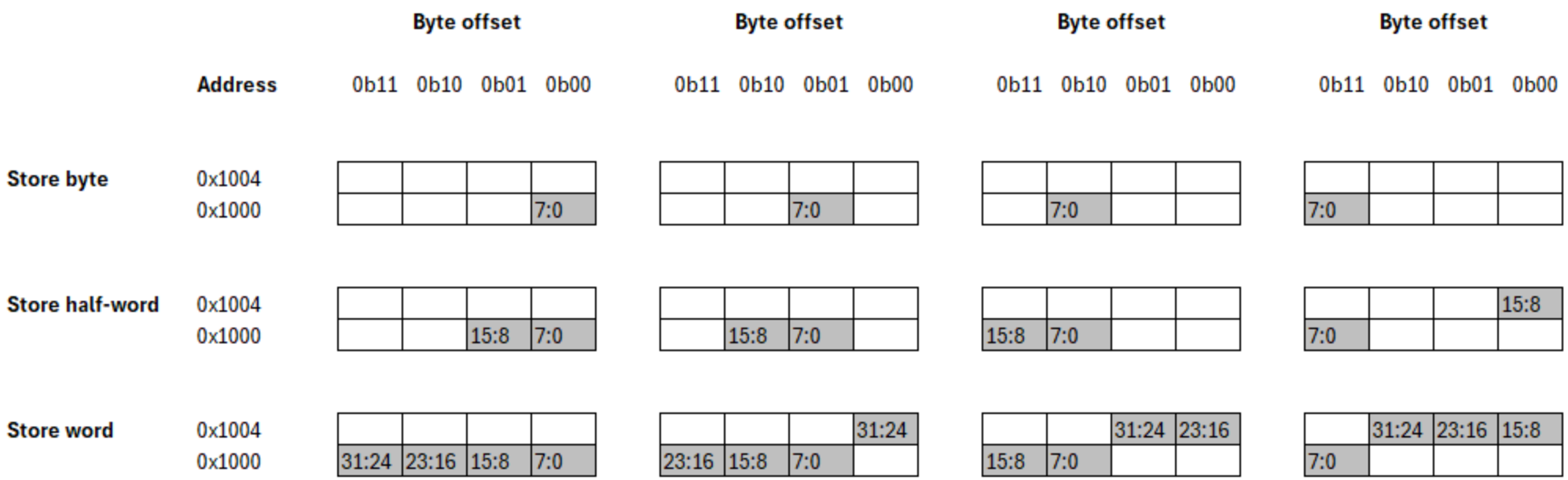
mem_to_reg	
Value	**Source**
00	alu_out
01	data_read_value
10	pc + 4
11	

branch_cond	
Value	**Description**
000	a == b
001	a != b
010	no branch
011	always branch
100	signed(a) < signed(b)
101	signed(a) >= signed(b)
110	a < b
111	a >= b

data_size	
Value	**Description**
000	byte
001	half-word
010	word
011	
100	unsigned byte
101	unsigned half-word
110	
111	

ALU Op					
opcode	**funct3**	**funct7[5]**	**Operation**	**Function**	**ALU Op**
0?1_0011	000	0	add	alu_out = a + b	0000
0?1_0011	000	1	sub	alu_out = a - b	1000
0?1_0011	001	0	sll	alu_out = a << b[4:0]	0001
0?1_0011	010	0	slt	alu_out = a < signed(b) ? 1 : 0	0010
0?1_0011	011	0	sltu	alu_out = a < unsigned(b) ? 1 : 0	0011
0?1_0011	100	0	xor	alu_out = a ^ b	0100
0?1_0011	101	0	srl	alu_out = a >> b[4:0]	0101
0?1_0011	101	1	sra	alu_out = a >>> b[4:0]	1101
0?1_0011	110	0	or	alu_out = a | b	0110
0?1_0011	111	0	and	alu_out = a & b	0111
				alu_out = b	1001

opcode		
Value	**a**	**b**
001_0011	rs1	imm12
011_0011	rs1	rs2


Byte offset
Byte offset
Byte offset
Byte offset
Address
0b11 0b10 0b01 0b00
0b11 0b10 0b01 0b00
0b11 0b10 0b01 0b00
0b11 0b10 0b01 0b00
Store byte
0x1004
0x1000
7:0
7:0
7:0
7:0
Store half-word
0x1004
0x1000
15:8
7:0
15:8
7:0
15:8
7:0
15:8
7:0
Store word
0x1004
0x1000
31:24
23:16
15:8
7:0
31:24
23:16
15:8
7:0
31:24
23:16
15:8
7:0
31:24
23:16
15:8
7:0