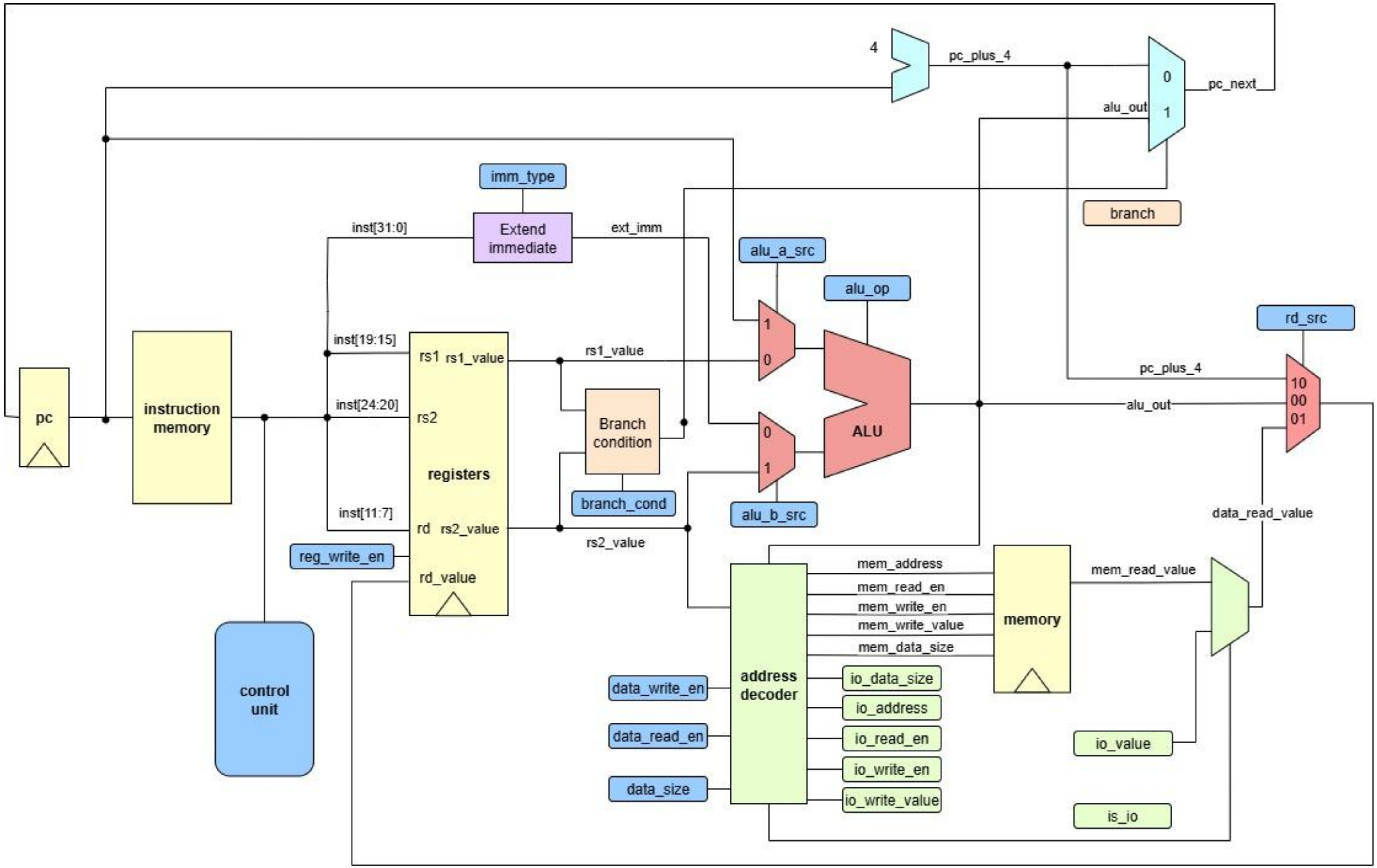
RISC V


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

<table>
<tr><td></td><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td></td></tr>
<tr><td>Load</td><td>I</td><td colspan="12">imm[11:0]</td><td colspan="5">rs1</td><td colspan="3">funct3</td><td colspan="5">rd</td><td colspan="7">op</td><td>12</td></tr>
<tr><td>Store</td><td>S</td><td colspan="7">imm[11:5]</td><td colspan="5">rs2</td><td colspan="5">rs1</td><td colspan="3">funct3</td><td colspan="5">imm[4:0]</td><td colspan="7">op</td><td>12</td></tr>
<tr><td>Data</td><td>R</td><td colspan="7">funct7</td><td colspan="5">rs2</td><td colspan="5">rs1</td><td colspan="3">funct3</td><td colspan="5">rd</td><td colspan="7">op</td><td></td></tr>
<tr><td>Load imm</td><td>U</td><td colspan="20">imm[31:12]</td><td colspan="5">rd</td><td colspan="7">op</td><td>20</td></tr>
<tr><td>Branch</td><td>B</td><td>i[12]</td><td colspan="6">imm[10:5]</td><td colspan="5">rs2</td><td colspan="5">rs1</td><td colspan="3">funct3</td><td colspan="4">imm[4:1]</td><td>i[11]</td><td colspan="7">op</td><td>12</td></tr>
<tr><td>Jump</td><td>J</td><td>i[20]</td><td colspan="10">imm[10:1]</td><td>i[11]</td><td colspan="8">imm[19:12]</td><td colspan="5">rd</td><td colspan="7">op</td><td>20</td></tr>
</table>

<table>
<tr><td></td><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Load</td><td>I</td><td colspan="21">inst[31]</td><td colspan="6">inst[30:25]</td><td colspan="4">inst[24:21]</td><td>i[20]</td></tr>
<tr><td>Store</td><td>S</td><td colspan="21">inst[31]</td><td colspan="6">inst[30:25]</td><td colspan="4">inst[11:8]</td><td>i[7]</td></tr>
<tr><td>Load imm</td><td>U</td><td>i[31]</td><td colspan="11">inst[30:20]</td><td colspan="8">inst[19:12]</td><td colspan="12">0</td></tr>
<tr><td>Branch</td><td>B</td><td colspan="20">inst[31]</td><td>i[7]</td><td colspan="6">inst[30:25]</td><td colspan="4">inst[11:8]</td><td>0</td></tr>
<tr><td>Jump</td><td>J</td><td colspan="12">inst[31]</td><td colspan="8">imm[19:12]</td><td>i[20]</td><td colspan="6">inst[30:25]</td><td colspan="4">inst[24:21]</td><td>0</td></tr>
</table>

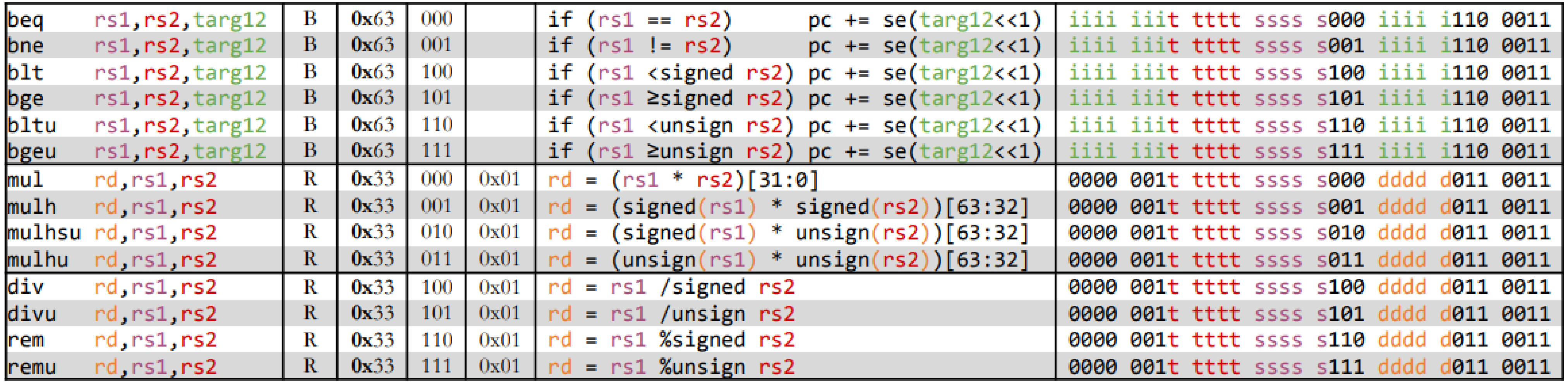
<table>
<tr><td>31</td><td>30 25</td><td>24 21</td><td>20</td><td>19 15</td><td>14 12</td><td>11 8</td><td>7</td><td>6 0</td><td></td></tr>
<tr><td colspan="2">funct7</td><td colspan="2">rs2</td><td>rs1</td><td>funct3</td><td colspan="2">rd</td><td>opcode</td><td>R-type</td></tr>
<tr><td colspan="4">imm[11:0]</td><td>rs1</td><td>funct3</td><td colspan="2">rd</td><td>opcode</td><td>I-type</td></tr>
<tr><td colspan="2">imm[11:5]</td><td colspan="2">rs2</td><td>rs1</td><td>funct3</td><td colspan="2">imm[4:0]</td><td>opcode</td><td>S-type</td></tr>
<tr><td>imm[12]</td><td>imm[10:5]</td><td colspan="2">rs2</td><td>rs1</td><td>funct3</td><td>imm[4:1]</td><td>imm[11]</td><td>opcode</td><td>B-type</td></tr>
<tr><td colspan="6">imm[31:12]</td><td colspan="2">rd</td><td>opcode</td><td>U-type</td></tr>
<tr><td>imm[20]</td><td colspan="2">imm[10:1]</td><td>imm[11]</td><td colspan="2">imm[19:12]</td><td colspan="2">rd</td><td>opcode</td><td>J-type</td></tr>
</table>

<table>
<tr><td>31</td><td>30 20</td><td>19 12</td><td>11</td><td>10 5</td><td>4 1</td><td>0</td><td></td></tr>
<tr><td colspan="4">— inst[31] —</td><td>inst[30:25]</td><td>inst[24:21]</td><td>inst[20]</td><td>I-immediate</td></tr>
<tr><td colspan="4">— inst[31] —</td><td>inst[30:25]</td><td>inst[11:8]</td><td>inst[7]</td><td>S-immediate</td></tr>
<tr><td colspan="3">— inst[31] —</td><td>inst[7]</td><td>inst[30:25]</td><td>inst[11:8]</td><td>0</td><td>B-immediate</td></tr>
<tr><td>inst[31]</td><td>inst[30:20]</td><td>inst[19:12]</td><td colspan="4">— 0 —</td><td>U-immediate</td></tr>
<tr><td colspan="2">— inst[31] —</td><td>inst[19:12]</td><td>inst[20]</td><td>inst[30:25]</td><td>inst[24:21]</td><td>0</td><td>J-immediate</td></tr>
</table>

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

Inst Type	Bits	Hex max	Unsigned max	Bit Range	Instruction Range
I	12	0x00 00 0F FF	4095	-2048 to 2047	-2048 to 2047
S	12	0x00 00 0F FF	4095	-2048 to 2047	-2048 to 2047
U	20	0x00 0F FF FF	1 048 575	-524 288 to 524 287 0 to 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	4095	-2048 to 2047	-4096 to 4095 in steps of 2
J	20	0x00 0F FF FF	1 048 575	-524 288 to 524 287	-1 048 576 to 1 048 575 in steps of 2

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	00	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