7/29 MEETING

Outline

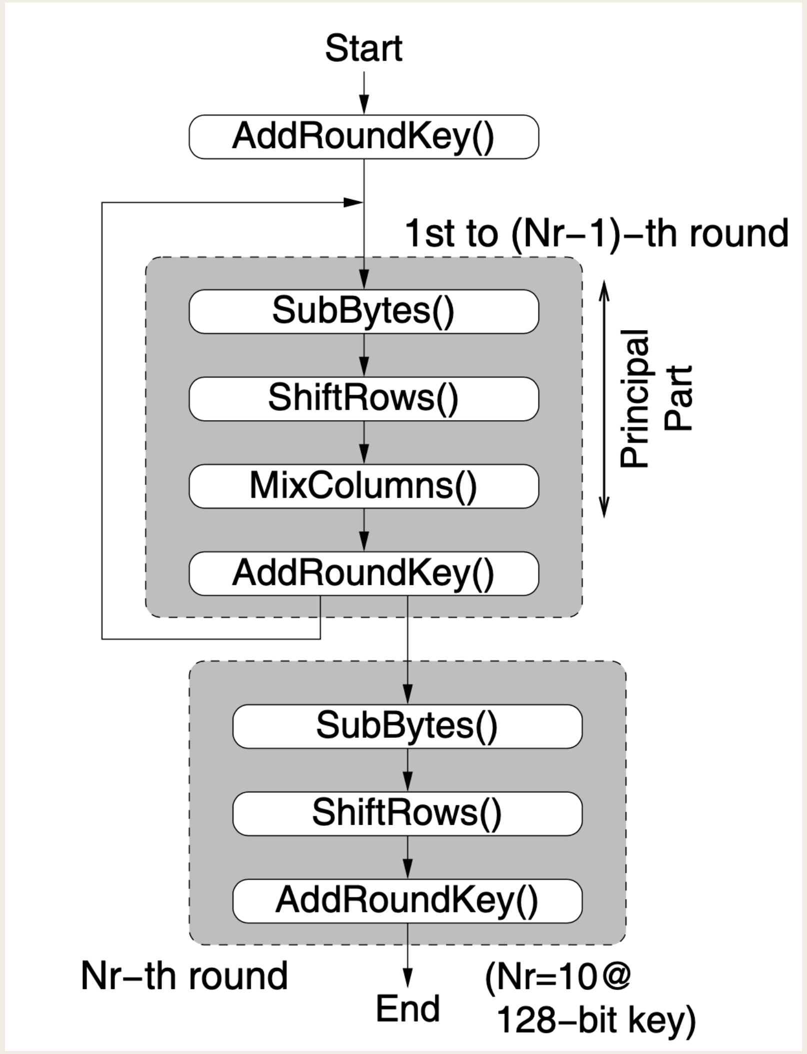
1. Paper reading
2. Turn RV64I to RV32I and modify tb
3. Add new instructions (saes, auipc, mul…)
4. Generate AES pattern (encrypt+decrypt)
5. Simulate and debug
6. Practice Summer_Training_v2.0

Paper reading

1. The design of scalar AES Instruction Set Extensions for RISC-V (MARSHALL, Ben, et al.)
2. The Design of Rijndael (springer, 2nd edition)
3. A Lightweight ISA Extension for AES and SM4.(Saarinen, Markku-Juhani O.)
4. A depth-16 circuit for the AES S-box (Boyar, Joan, and René Peralta)
5. Extended instructions for the AES cryptography and their efficient implementation (Nadehara, Kouhei, Masao Ikekawa, and Ichiro Kuroda)

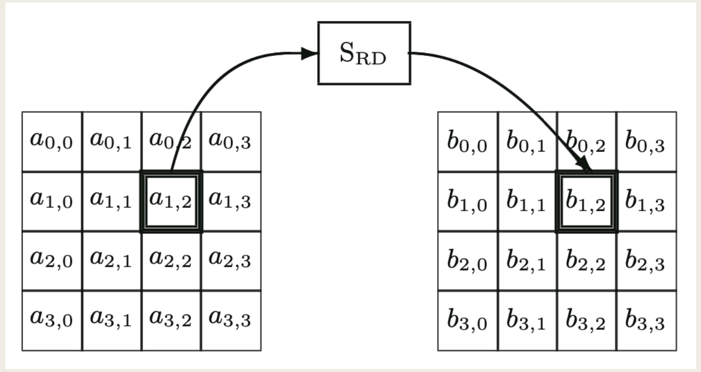
AES (advanced encryption standard)

- AES-128/192/256:
  - Plain text : 128bits (16 bytes)
  - Key : 128/192/256bits
  - Rounds : 10/12/14
  - Cypher : 128bits (16 bytes)



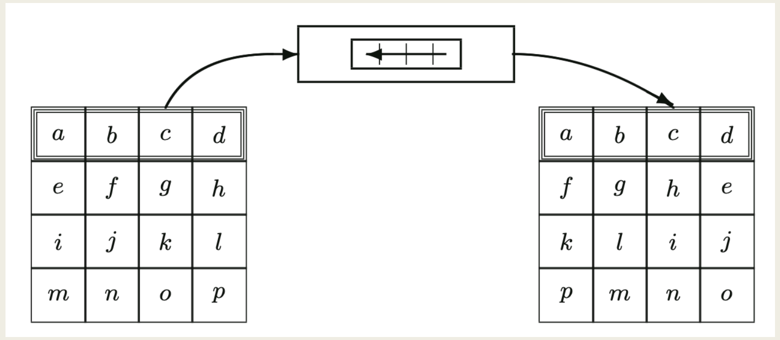
AES (advanced encryption standard)

- SubBytes (S-Box)

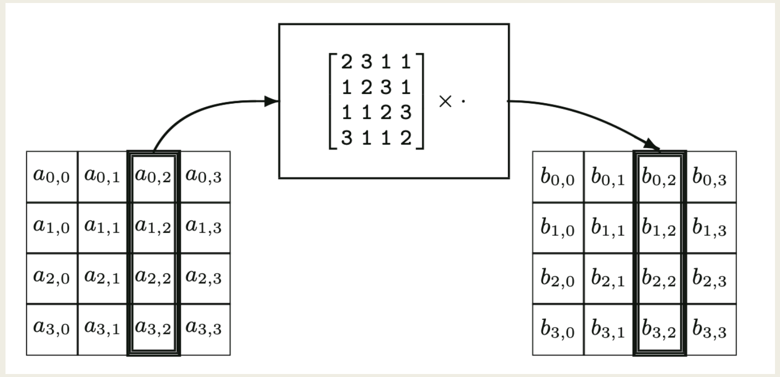


S_RD
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

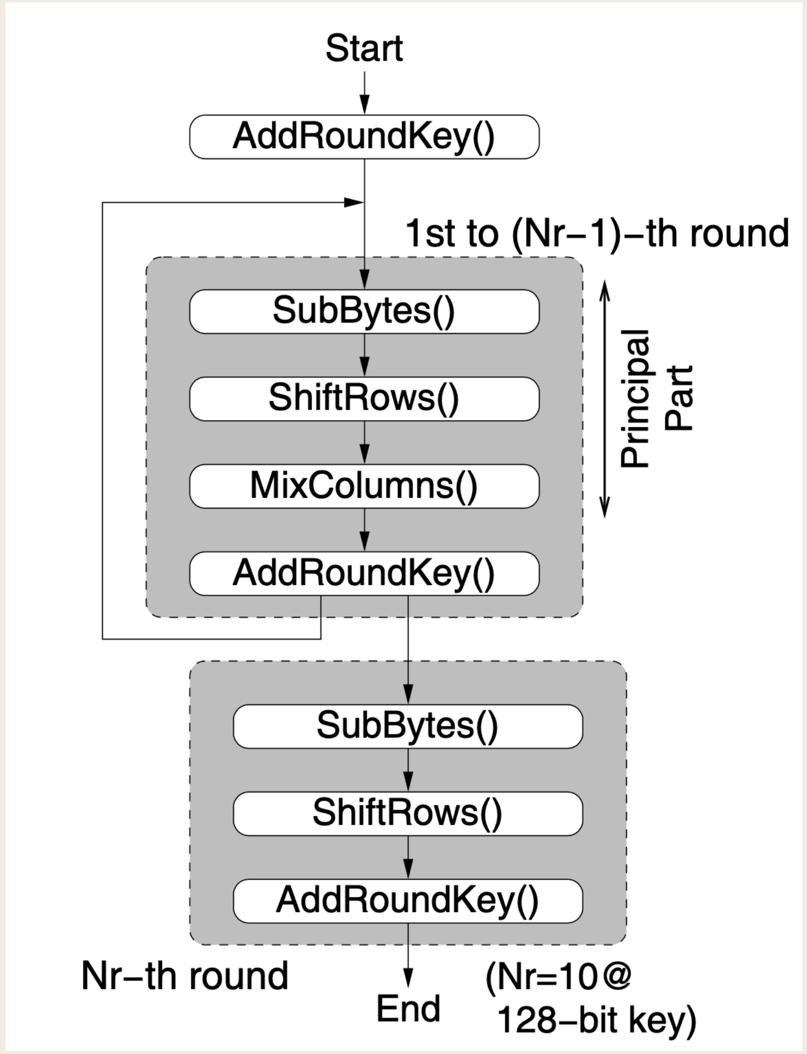
- ShiftRows



- MixColumns



$\begin{bmatrix}2&3&1&1\\1&2&3&1\\1&1&2&3\\3&1&1&2\end{bmatrix} \times \cdot$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$



AES (advanced encryption standard)

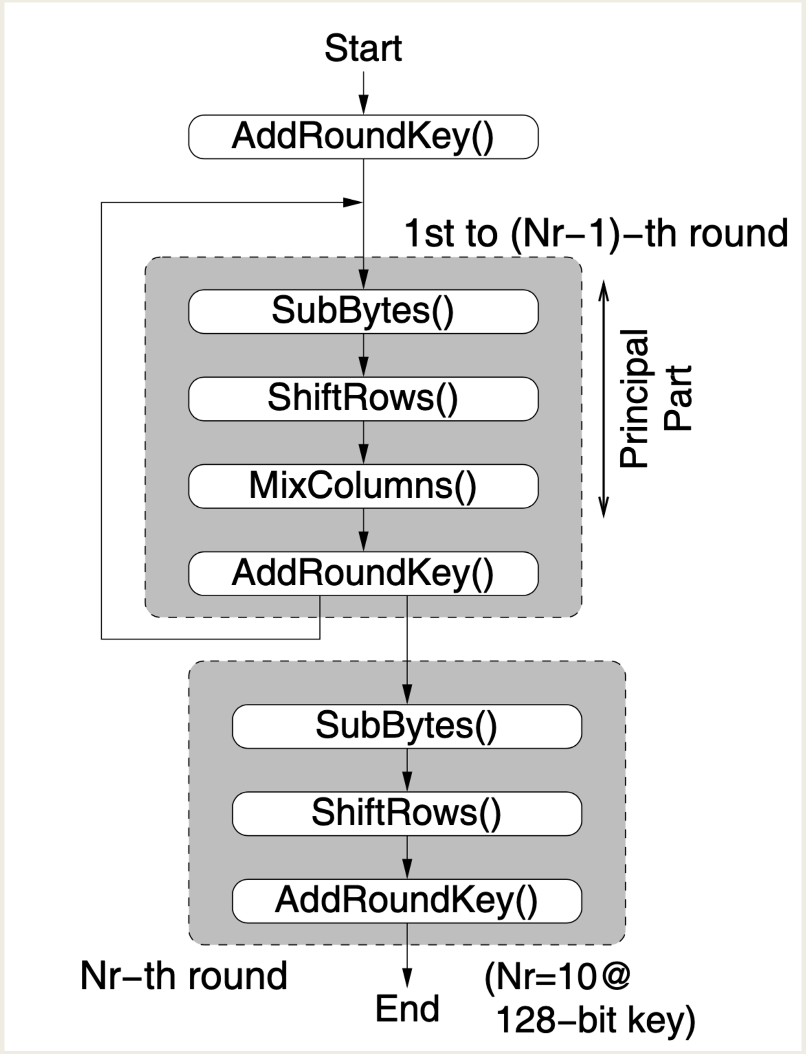
- AddRoundKey

$$\begin{array}{|c|c|c|c|}\hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ \hline a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ \hline a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array} + \begin{array}{|c|c|c|c|}\hline k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ \hline k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ \hline k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ \hline k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ \hline b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ \hline \end{array}$$

- Keyexpansion
  - Nk: key block number
  - Nb: pt block number

Ex: Nk=Nb=4, rounds=10, roundkey=16*11bytes

	N_b				
N_k	4	5	6	7	8
4	10	11	12	13	14
5	11	11	12	13	14
6	12	12	12	13	14
7	13	13	13	13	14
8	14	14	14	14	14



Implement AES

- Compute-oriented
  - Online computation, less memory
  - Substitute: pre-compute S-BOX /xtime, less latency
- Table-oriented
  - Offline pre-computation, less latency but increasing memory
  - Column-packed
- Bit-sliced
  - Avoid timing attacks on processors with cache memory due to table look-ups.

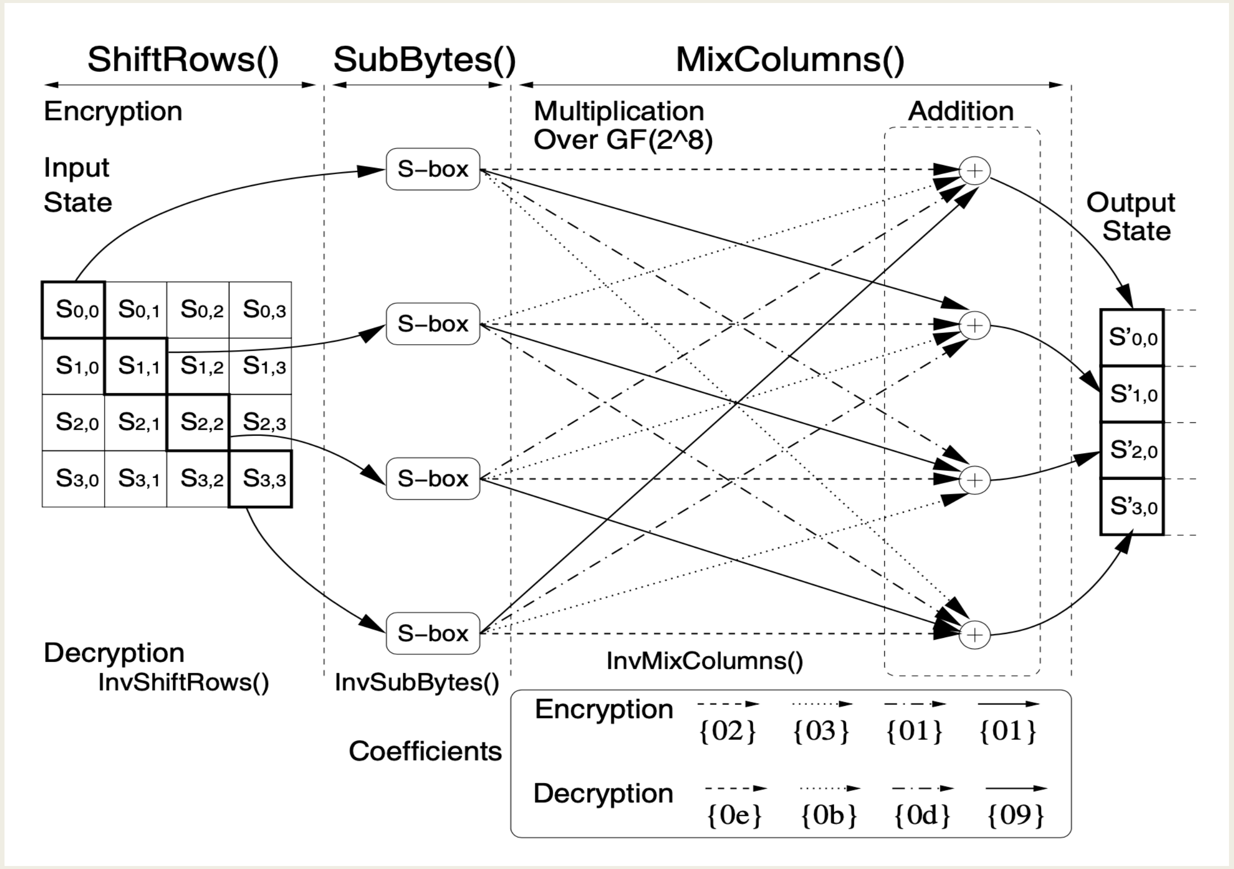
T-table

- Linear combination of four column vectors by matrix multiplications.
- One byte in one operation.
- Using instructions to choose

bytes.

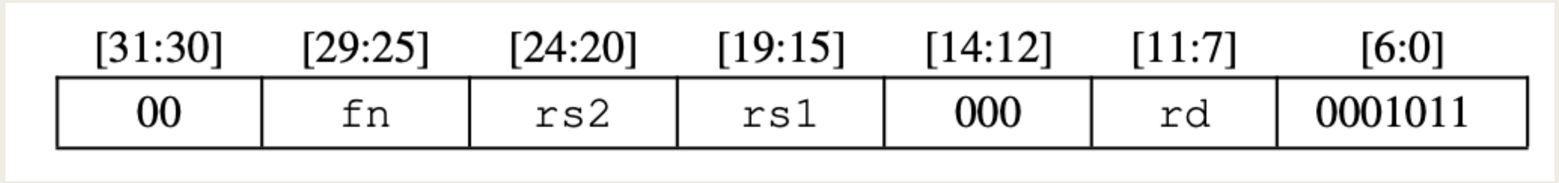
$$T_0[x] = \begin{bmatrix} 02_{(16)} \otimes \text{S-BOX}(x) \\ 01_{(16)} \otimes \text{S-BOX}(x) \\ 01_{(16)} \otimes \text{S-BOX}(x) \\ 03_{(16)} \otimes \text{S-BOX}(x) \end{bmatrix} \qquad T_1[x] = \begin{bmatrix} 03_{(16)} \otimes \text{S-BOX}(x) \\ 02_{(16)} \otimes \text{S-BOX}(x) \\ 01_{(16)} \otimes \text{S-BOX}(x) \\ 01_{(16)} \otimes \text{S-BOX}(x) \end{bmatrix}$$

$$T_2[x] = \begin{bmatrix} 01_{(16)} \otimes \text{S-BOX}(x) \\ 03_{(16)} \otimes \text{S-BOX}(x) \\ 02_{(16)} \otimes \text{S-BOX}(x) \\ 01_{(16)} \otimes \text{S-BOX}(x) \end{bmatrix} \qquad T_3[x] = \begin{bmatrix} 01_{(16)} \otimes \text{S-BOX}(x) \\ 01_{(16)} \otimes \text{S-BOX}(x) \\ 03_{(16)} \otimes \text{S-BOX}(x) \\ 02_{(16)} \otimes \text{S-BOX}(x) \end{bmatrix}$$



Hardware-assisted T-table

- https://github.com/mjosaarinen/lwaes_isa
- Add 4 instructions (2 for encryption, 2 for decryption)
- Encryption requires 20 instructions/round (4 lw, 16 saes.v3.encsm)

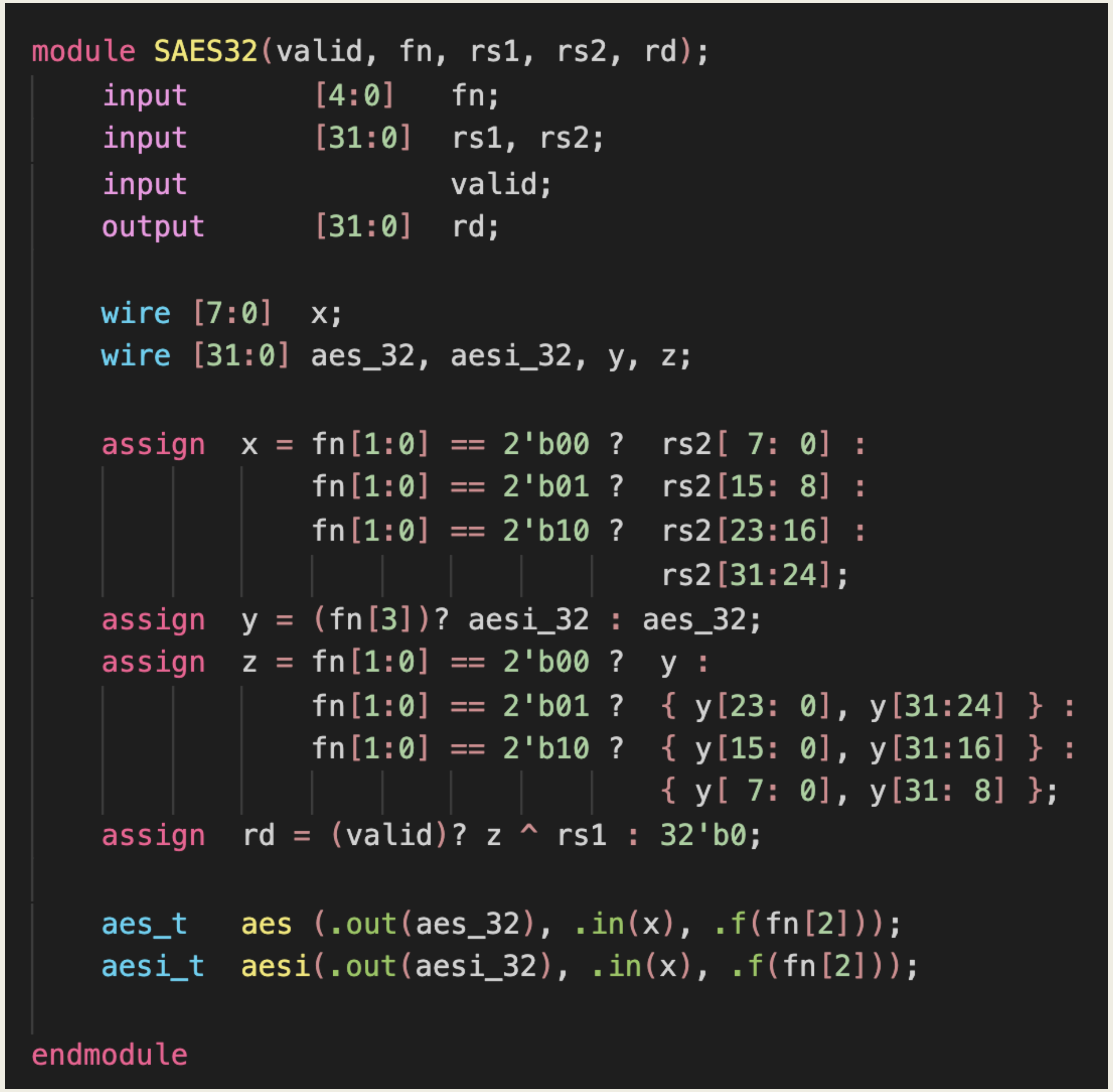


[31:30]	[29:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]
00	fn	rs2	rs1	000	rd	0001011

- fn [1:0]: select a single byte from rs2.
- fn [4:2]: specify saes32.encsm/saes32.encs/saes32.decsm/saes32.decs
- rs1: store round key
- rs2: store cypher
- rs1=rd

Implement

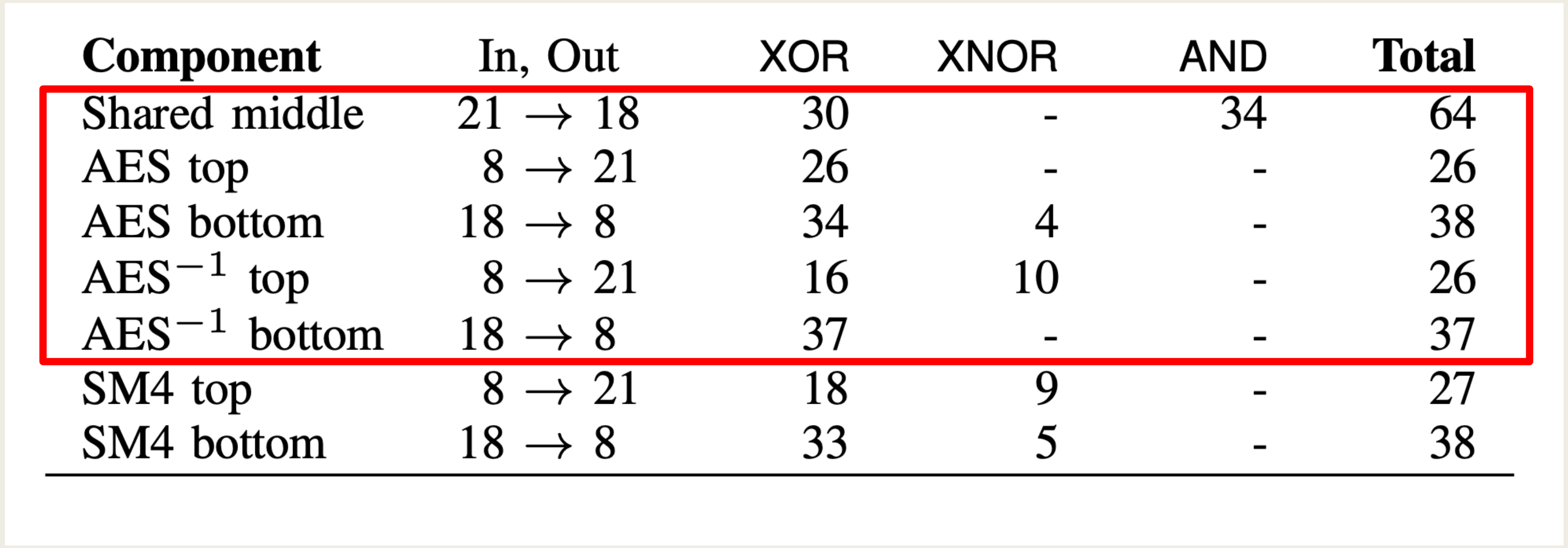
- 5 stages RISC-V 32bit
- Add AES signal in *control_unit (2nd stage)*
- Add *saes32_unit (3rd stage)*



```
module SAES32(valid, fn, rs1, rs2, rd);
    input       [4:0]   fn;
    input       [31:0]  rs1, rs2;
    input               valid;
    output      [31:0]  rd;

    wire [7:0]  x;
    wire [31:0] aes_32, aesi_32, y, z;

    assign  x = fn[1:0] == 2'b00 ?  rs2[ 7: 0] :
                fn[1:0] == 2'b01 ?  rs2[15: 8] :
                fn[1:0] == 2'b10 ?  rs2[23:16] :
                                    rs2[31:24];
    assign  y = (fn[3])? aesi_32 : aes_32;
    assign  z = fn[1:0] == 2'b00 ?  y :
                fn[1:0] == 2'b01 ?  { y[23: 0], y[31:24] } :
                fn[1:0] == 2'b10 ?  { y[15: 0], y[31:16] } :
                                    { y[ 7: 0], y[31: 8] };
    assign  rd = (valid)? z ^ rs1 : 32'b0;

    aes_t   aes (.out(aes_32), .in(x), .f(fn[2]));
    aesi_t  aesi(.out(aesi_32), .in(x), .f(fn[2]));

endmodule
```

S-box

- A depth-16 circuit for the AES S-box (Boyar, Joan, and René Peralta)
- Three layers: Top -> Middle -> Bottom
- AES S-box uses 128 gates
- AES^{-1} S-box uses 127 gates
- Total uses 191 gates



Component	In, Out	XOR	XNOR	AND	**Total**
Shared middle	$21 \rightarrow 18$	30	-	34	64
AES top	$8 \rightarrow 21$	26	-	-	26
AES bottom	$18 \rightarrow 8$	34	4	-	38
AES^{-1} top	$8 \rightarrow 21$	16	10	-	26
AES^{-1} bottom	$18 \rightarrow 8$	37	-	-	37
SM4 top	$8 \rightarrow 21$	18	9	-	27
SM4 bottom	$18 \rightarrow 8$	33	5	-	38

Generate pattern

- aes.encsm.s (including key expansion and encryption)
  - Input: plain text, key 128bits
  - Output: round key, cypher
- aes.decsm.s (including key expansion and decryption)
  - Input: plain text, key 128bits
  - Output: round key, plain text
- Decode assembly to machine code
  - Use Jupiter RISCV-simulator
  - Own python file (for ISA)

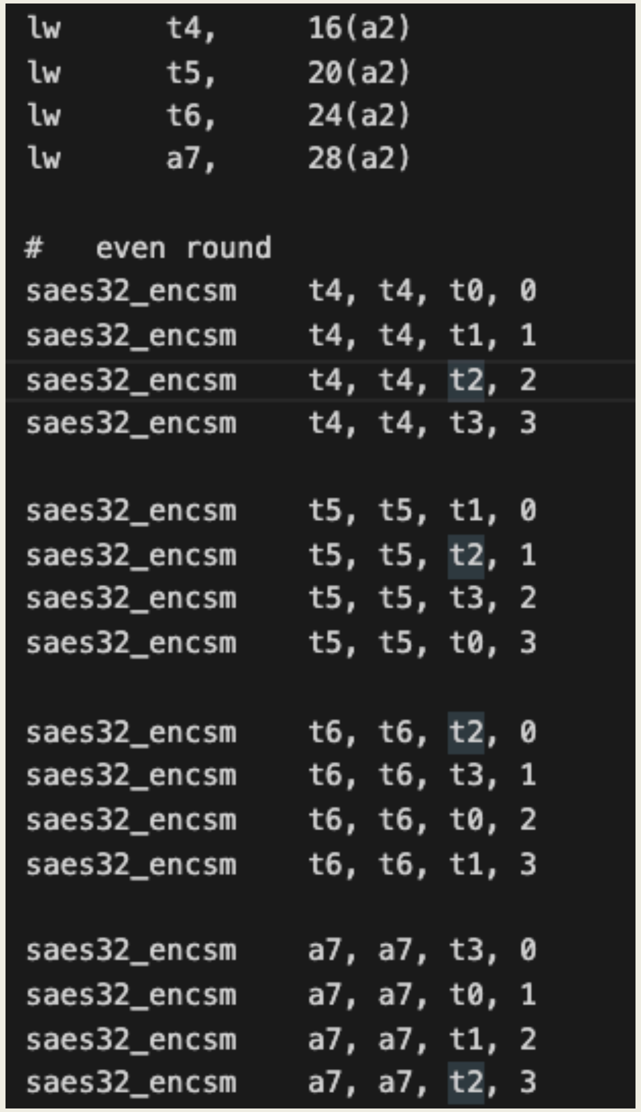
Generate pattern

saes32_encsm rd rs1 rs2 bt

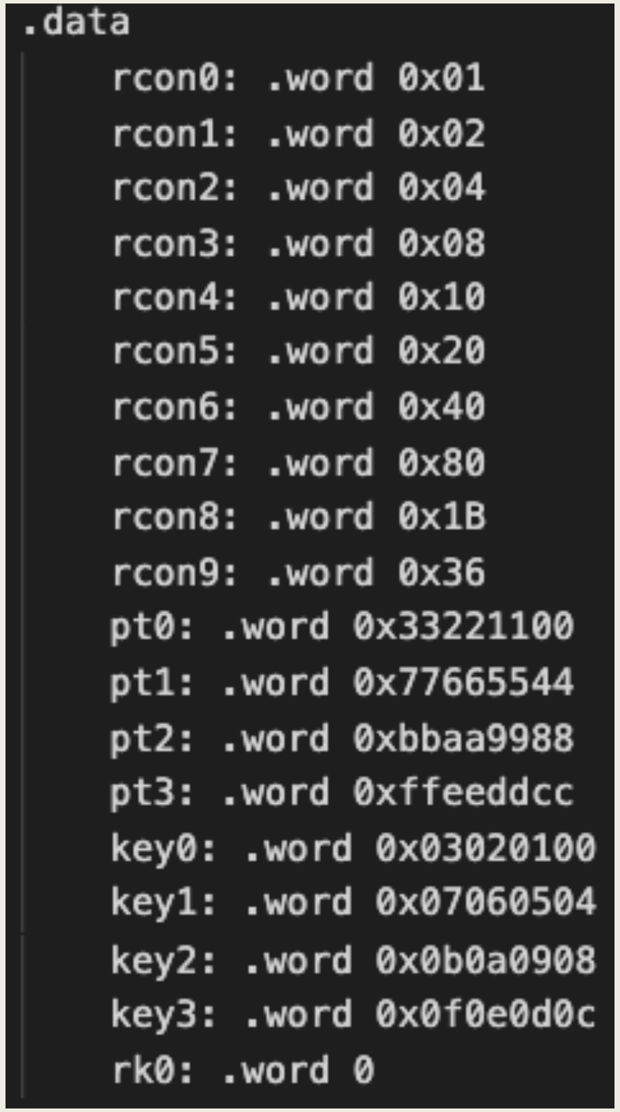
saes32_encs rd rs1 rs2 bt

saes32_decsm rd rs1 rs2 bt

saes32_decs rd rs1 rs2 bt



```
lw      t4,     16(a2)
lw      t5,     20(a2)
lw      t6,     24(a2)
lw      a7,     28(a2)

#   even round
saes32_encsm    t4, t4, t0, 0
saes32_encsm    t4, t4, t1, 1
saes32_encsm    t4, t4, t2, 2
saes32_encsm    t4, t4, t3, 3

saes32_encsm    t5, t5, t1, 0
saes32_encsm    t5, t5, t2, 1
saes32_encsm    t5, t5, t3, 2
saes32_encsm    t5, t5, t0, 3

saes32_encsm    t6, t6, t2, 0
saes32_encsm    t6, t6, t3, 1
saes32_encsm    t6, t6, t0, 2
saes32_encsm    t6, t6, t1, 3

saes32_encsm    a7, a7, t3, 0
saes32_encsm    a7, a7, t0, 1
saes32_encsm    a7, a7, t1, 2
saes32_encsm    a7, a7, t2, 3
```



```
.data
    rcon0: .word 0x01
    rcon1: .word 0x02
    rcon2: .word 0x04
    rcon3: .word 0x08
    rcon4: .word 0x10
    rcon5: .word 0x20
    rcon6: .word 0x40
    rcon7: .word 0x80
    rcon8: .word 0x1B
    rcon9: .word 0x36
    pt0: .word 0x33221100
    pt1: .word 0x77665544
    pt2: .word 0xbbaa9988
    pt3: .word 0xffeeddcc
    key0: .word 0x03020100
    key1: .word 0x07060504
    key2: .word 0x0b0a0908
    key3: .word 0x0f0e0d0c
    rk0: .word 0
```

Complete

1. AES encryption (for RV32I)
2. AES decryption (for RV32I)
3. Pattern testing pass!