Assembly Code

Draft 11/25

# **RC5 Encryption**

**START:** SUB R2, R2, R2 \\ LOOP COUNTER = 0

SUB R3, R3, R3 \\ S ARRAY START ADDRESS = 0

SUB R11, R11, R11 \\ CLEAR LOOP COUNTER MAX

ADDi R11, R11, 0x000C \\ LOOP COUNTER MAX = 12

LW R3, R0, 0x0000 \\ ASSUMING THAT A IS STORED AT THE START OF THE MEMORY

LW R3, R1, 0x0001 \\ ASSUMING THAT B IS STORED AT LOCATION 1 OF THE MEMORY

LW R3, R4, 0x0006 \\ ASSUMING THAT S ARRAY IS STORED STARTING FROM LOCATION 6 OF THE MEMORY

ADD R0, R4, R0 \\ A = A + S[0]

LW R3, R4, 0x0007 \\ ASSUMING THAT S ARRAY IS STORED STARTING FROM LOCATION 6 OF THE MEMORY

ADD R1, R4, R1 \\ B = B + S[1]

ADD R3, R3, 0x0007 \\ R3 POINTS TO S ARRAY [1] (DATA MEM: A, B, S[0], S[1], S[2]….)

**LOOP:** SUB R5, R5, R5 \\ CLEAR R5

ADD R1, R5, R5 \\ R5 HAS COPY OF R1 WHICH IS VALUE OF B

NOR R5, R5, R5 \\ R5 HAS B BAR

SUB R6, R6, R6 \\ CLEAR R6

ADD R0, R6, R6 \\ R6 HAS COPY OF R0 WHICH IS VALUE OF A

NOR R6, R6, R6 \\ R6 HAS A BAR

NOR R5, R0, R9 \\ R9 HAS ABAR.B

NOR R6, R1, R5 \\ R5 NOW HAS BBAR.A

OR R0, R5, R9 \\ R9 NOW HAS ABAR.B + BBAR.A = A XOR B

SUB R8, R8, R8 \\ CLEAR ROTATE LOOP COUNTER.

**ROTATE\_LOOP:**

SUB R7, R7, R7 \\ CLEAR R7

ADD R7, R9, R7 \\ R7 ALSO HOLDS A\_XOR

SHL R9, R9, 0x0001 \\ SHIFT A\_XOR LEFT BY 1

SHR R7, R7, 0xFFFF \\ SHIFT A\_XOR RIGHT BY 1

ADD R7, R9, R0 \\ NEW VALUE OF A IS IN R0 WHICH IS THE ROTATE VALUE OF A\_XOR

ADD R8, R8, 0x0001 \\ INCREMENT ROTATE LOOP COUNTER.

BNE R8, R1, **ROTATE\_LOOP** \\ ROTATE BY REPEATED SINGLE SHIFT + ADD. REPEATS B NUMBER OF TIMES

LW R3, R4, 0x0001 \\ LOAD S ARRAY [2]. REMEMBER R3 POINTED TO S[1]. OFFSET HERE IS 1. SO R4 GETS S[2]

ADD R3, R3, 0X0001 \\ R3 POINTS TO NEXT S ARRAY ELEMENT

ADD R0, R4, R0 \\ FINAL VALUE OF A IN THIS ROUND. A = A\_ROT + S[2\*i]

SUB R5, R5, R5 \\ CLEAR R5

ADD R1, R5, R5 \\ R5 HAS COPY OF R1 WHICH IS VALUE OF B

NOR R5, R5, R5 \\ R5 HAS B BAR

SUB R6, R6, R6 \\ CLEAR R6

ADD R0, R6, R6 \\ R6 HAS COPY OF R0 WHICH IS VALUE OF UPDATED A

NOR R6, R6, R6 \\ R6 HAS A BAR

NOR R5, R0, R9 \\ R9 HAS ABAR.B

NOR R6, R1, R5 \\ R5 NOW HAS BBAR.A

OR R0, R5, R9 \\ R9 NOW HAS ABAR.B + BBAR.A = A XOR B

SUB R8, R8, R8 \\ CLEAR ROTATE LOOP COUNTER.

**ROTATE\_LOOP:**

SUB R7, R7, R7 \\ CLEAR R7

ADD R7, R9, R7 \\ R7 ALSO HOLDS B\_XOR

SHL R9, R9, 0x0001 \\ SHIFT B\_XOR LEFT BY 1

SHR R7, R7, 0xFFFF \\ SHIFT B\_XOR RIGHT BY 1

ADD R7, R9, R1 \\ NEW VALUE OF B IS IN R1 WHICH IS THE ROTATE VALUE OF B\_XOR

ADDi R8, R8, 0x0001 \\ INCREMENT ROTATE LOOP COUNTER.

BNE R8, R0, **ROTATE\_LOOP** \\ ROTATE BY REPEATED SINGLE SHIFT + ADD. REPEATS A NUMBER OF TIMES

LW R3, R4, 0x0001 \\ LOAD S ARRAY [2\*i + 1]. R3 WAS UPDATED IN PREVIOUS SECTION.

ADDi R3, R3, 0x0001 \\ R3 POINTS TO NEXT S ARRAY ELEMENT

ADD R1, R4, R1 \\ FINAL VALUE OF B IN THIS ROUND. B = B\_ROT + S[2\*i + 1]

SUB R8, R8, R8 \\ CLEAR ROTATE LOOP COUNTER.

ADDi R2, R2, 0x0001 \\ INCREMENT OUTTER LOOP COUNTER

BNE R2, R11, **LOOP**

HALT \\ OUTPUT AVAILABLE AT R0 AND R1 (A AND B)

# **RC5 Decryption**

**START:** SUB R2, R2, R2 \\ LOOP COUNTER = 0

SUB R3, R3, R3 \\ S ARRAY START ADDRESS = 0

SUB R11, R11, R11 \\ CLEAR LOOP COUNTER MAX

ADDi R11, R11, 0x000C \\ LOOP COUNTER MAX = 12

LW R3, R0, 0x0000 \\ ASSUMING THAT A IS STORED AT THE START OF THE MEMORY

LW R3, R1, 0x0001 \\ ASSUMING THAT B IS STORED AT LOCATION 1 OF THE MEMORY

ADD R3, R3, 0x001E\\ R3 POINTS TO S ARRAY [24] (DATA MEM: A, B, S[0], S[1], S[2]….) 1E (HEX) (INITIAL VALUE)

**LOOP:** LW R3, R4, 0x0001 \\ ASSUMING THAT S ARRAY IS STORED STARTING FROM LOCATION 31 OF THE MEMORY

SUB R10, R4, R1 \\ R1 HAS B = B - S[2\*I + 1]

**ROTATE\_LOOP:**

SUB R7, R7, R7 \\ CLEAR R7

ADD R7, R1, R7 \\ R7 ALSO HOLDS B\_SUB

SHR R9, R9, 0x0001 \\ SHIFT B\_SUB RIGHT BY 1

SHL R7, R7, 0xFFFF \\ SHIFT B\_SUB LEFT BY 1

ADD R7, R9, R1 \\ NEW VALUE OF B IS IN R1 WHICH IS THE ROTATE VALUE OF B\_SUB

ADD R8, R8, 0x0001 \\ INCREMENT ROTATE LOOP COUNTER.

BNE R8, R0, **ROTATE\_LOOP** \\ ROTATE BY REPEATED SINGLE SHIFT + ADD. REPEATS A NUMBER OF TIMES

SUB R8, R8, R8 \\ CLEAR ROTATE LOOP COUNTER.

SUB R5, R5, R5 \\ CLEAR R5

ADD R1, R5, R5 \\ R5 HAS COPY OF R1 WHICH IS VALUE OF B

NOR R5, R5, R5 \\ R5 HAS B BAR

SUB R6, R6, R6 \\ CLEAR R6

ADD R0, R6, R6 \\ R6 HAS COPY OF R0 WHICH IS VALUE OF A

NOR R6, R6, R6 \\ R6 HAS A BAR

NOR R5, R0, R9 \\ R9 HAS ABAR.B

NOR R6, R1, R5 \\ R5 NOW HAS BBAR.A

OR R9, R5, R1 \\ R1 NOW HAS ABAR.B + BBAR.A = A XOR B

SUB R3, R3, 0x0001 \\ R3 POINTS TO NEXT S ARRAY ELEMENT

LW R3, R4, 0x0001 \\ LOAD NEXT S ARRAY ELEMENT INTO R4

SUB R9, R4, R0 \\ R0 HAS A= A - S[2\*i]

**ROTATE\_LOOP:**

SUB R7, R7, R7 \\ CLEAR R7

ADD R7, R0, R7 \\ R7 ALSO HOLDS A\_SUB

SHR R9, R9, 0x0001 \\ SHIFT A\_SUB RIGHT BY 1

SHL R7, R7, 0xFFFF \\ SHIFT A\_SUB LEFT BY 1

ADD R7, R9, R0 \\ NEW VALUE OF A IS IN R1 WHICH IS THE ROTATE VALUE OF A\_SUB

ADD R8, R8, 0x0001 \\ INCREMENT ROTATE LOOP COUNTER.

BNE R8, R0, **ROTATE\_LOOP** \\ ROTATE BY REPEATED SINGLE SHIFT + ADD. REPEATS B NUMBER OF TIMES

SUB R8, R8, R8 \\ CLEAR ROTATE LOOP COUNTER.

SUB R5, R5, R5 \\ CLEAR R5

ADD R1, R5, R5 \\ R5 HAS COPY OF R1 WHICH IS VALUE OF B

NOR R5, R5, R5 \\ R5 HAS B BAR

SUB R6, R6, R6 \\ CLEAR R6

ADD R0, R6, R6 \\ R6 HAS COPY OF R0 WHICH IS VALUE OF A

NOR R6, R6, R6 \\ R6 HAS A BAR

NOR R5, R0, R9 \\ R9 HAS ABAR.B

NOR R6, R1, R5 \\ R5 NOW HAS BBAR.A

OR R9, R5, R0 \\ R0 NOW HAS ABAR.B + BBAR.A = A XOR B

ADDi R2, R2, 0x0001 \\ INCREMENT OUTTER LOOP COUNTER

BNE R2, R11, **LOOP**

LW R3, R4, 0x0006 \\ ASSUMING THAT S ARRAY IS STORED STARTING FROM LOCATION 6 OF THE MEMORY

SUB R0, R4, R0 \\ A = A - S[0]

LW R3, R4, 0x0007 \\ ASSUMING THAT S ARRAY IS STORED STARTING FROM LOCATION 6 OF THE MEMORY

SUB R1, R4, R1 \\ B = B - S[1]

HALT \\ OUTPUT AVAILABLE AT R0 AND R1 (A AND B)

# **RC5 Key Expansion**

SUB R0, R0, R0 \\ CLEAR R0

SUB R1, R1, R1 \\ CLEAR R1

SUB R2, R2, R2 \\ CLEAR R2

SUB R3, R3, R3 \\ CLEAR R3

SUB R4, R4, R4 \\ CLEAR R4

SUB R5, R5, R5 \\ CLEAR R5 I COUNTER

SUB R6, R6, R6 \\ CLEAR R6 J COUNTER

SUB R7, R7, R7 \\ CLEAR R7

SUB R8, R8, R8 \\ CLEAR R8 ZERO REGISTER

SUB R9, R9, R9 \\ CLEAR R9

SUB R10, R10, R10 \\ CLEAR R9

SUB R11, R11, R11 \\ CLEAR R9

SUB R12, R12, R12 \\ CLEAR R12 ALWAYS HOLD 25D MAX OF I COUNT

SUB R13, R13, R13 \\ CLEAR R13 ALWAYS HOLD 3 MAX OF J COUNT

ADDi R12, R12, 0x0019 \\ R12 ALWAYS HOLD 3 MAX OF J COUNT

ADDi R13, R13, 0x0003 \\ R13 ALWAYS HOLD 3 MAX OF J COUNT

ADDi R7, R7, 0x4D \\ LOOP COUNTER INITIALISED TO 77

**LOOP:**

LW R5, R2, 0x06 \\ LOAD DATA\_MEM[6] WHICH IS S\_ARRAY[0] INTO R2

LW R6, R3, 0x02 \\ LOAD DATA\_MEM[2] WHICH IS USER\_KEY[0] INTO R3

ADD R0, R1, R4 \\ R4 GETS R0 + R1 (A+B). R4 WILL ALWAYS HOLD A+B

ADD R4, R2, R0 \\ R0 GETS S[COUNT] + A + B

**ROTATE\_LOOP:**

SUB R9, R9, R9 \\ CLEAR R9

ADD R9, R0, R9 \\ R9 ALSO HOLDS A

SUB R11, R11, R11 \\ CLEAR R11

ADD R11, R0, R11\\ R9 ALSO HOLDS B

SHL R11, R11, 0x0001 \\ SHIFT A LEFT BY 1

SHR R9, R9, 0xFFFF \\ SHIFT A RIGHT BY 31

ADD R9, R11, R0 \\ NEW VALUE OF A IS IN R0 WHICH IS THE ROTATE VALUE OF A

ADD R10, R10, 0x0001 \\ INCREMENT ROTATE LOOP COUNTER.

BNE R10, R12, **ROTATE\_LOOP** \\ ROTATE BY REPEATED SINGLE SHIFT + ADD. REPEATS A NUMBER OF TIMES

SUB R10, R10, R10 \\ CLEAR ROTATE LOOP COUNTER.

ADD R0, R1, R4 \\ PERFORMS A+B AND STORES IN R4

ADD R4, R3, R1 \\ R1 GETS L[COUNT] + A + B

**ROTATE\_LOOP:**

SUB R9, R9, R9 \\ CLEAR R7

ADD R9, R1, R9 \\ R7 ALSO HOLDS B

SUB R11, R11, R11 \\ CLEAR R11

ADD R11, R1, R11 \\ R11 ALSO HOLDS B

SHL R11, R11, 0x0001 \\ SHIFT A LEFT BY 1

SHR R9, R9, 0xFFFF \\ SHIFT A RIGHT BY 31

ADD R9, R11, R1 \\ NEW VALUE OF B IS IN R1 WHICH IS THE ROTATE VALUE OF B

ADD R10, R10, 0x0001 \\ INCREMENT ROTATE LOOP COUNTER.

BNE R10, R4, **ROTATE\_LOOP** \\ ROTATE BY REPEATED SINGLE SHIFT + ADD. REPEATS A+B NUMBER OF TIMES

SUB R10, R10, R10 \\ CLEAR ROTATE LOOP COUNTER.

SW R5, R0, 0x0000 \\ STORE A INTO DATA\_MEM POINTED BY R5 WHICH IS ACTUALLY VALUE OF A IN THIS ITERATION

SW R6, R1, 0x0000 \\ STORE B INTO DATA\_MEM POINTED BY R6 WHICH IS ACTUALLY VALUE OF B IN THIS ITERATION

BNE R5, R8, **Check\_J\_Count** \\ IF I COUNTER IS NOT EQUAL TO 25, CHECK J COUNTER. ELSE RESET I COUNTER

SUB R5, R5, R5 \\ RESET I COUNTER

**Check\_J\_Count:**

BNE R6, R8, **CONT**\\ IF J COUNTER IS NOT EQUAL TO 3, CONTINUE COMPUTATION. ELSE RESET J COUNTER

SUB R6, R6, R6 \\ RESET J COUNTER

**CONT:**

SUB R7,R7, 0x0001 \\ RESET I COUNTER DECREMENT K COUNTER

BNE R7, R8, **LOOP** \\ IF K COUNTER IS NOT EQUAL TO 0, THEN GO BACK TO LOOP. ELSE STOP.

HAL

# **S Array Generation**

SUB R0, R0, R0 \\ CLEAR R0

SUB R1, R1, R1 \\ CLEAR R1

SUB R2, R2, R2 \\ CLEAR R2

SUB R3, R3, R3 \\ CLEAR R3

ADDi R3, R3, 0x001A \\ R3 IS MAX VALUE OF COUNTER = 26.

ADDi R0, R0, 0x0006 \\ INITIAL ADDRESS IN DATA MEMORY WHERE S\_MATRIX IS STORED

ADDi R1, R1, 0xB7E1 \\ 16 BITS OF THE MAGIC NUMBER Pw

SHL R1, R1, 0x00FF \\ SHIFT ABOVE VALUE TO THE MSB LOCATIONS

ADDi R1, R1, 0x5163 \\ LOWER 16 BITS OF MAGIC NUMBER Pw

ADDi R2, R2, 0x9E37 \\ 16 BITS OF THE MAGIC NUMBER Qw

SHL R2, R2, 0x00FF \\ SHIFT ABOVE VALUE TO THE MSB LOCATIONS

ADDi R2, R2, 0x79B9 \\ LOWER 16 BITS OF MAGIC NUMBER Qw

**LOOP:**

SW R0, R1, 0x0000 \\ STORE GENERATED VALUE [R1] IN LOCATION POINTED BY R0

ADDi R0, R0, 0x0001 \\ INCREMENT R0

ADD R1, R2, R1 \\ PERFORM Pw + n\*Qw

BNE R0, R3, **LOOP** \\ IF COUNTER IS LESS THAN 26, LOOP. ELSE STOP.

HAL

# **APPENDIX**

## **Register file:**

*--R0: Stores 32 bit value of A*

*--R1: Stores 32 bit value of B*

*--R2: Loop counter*

*--R3: Data Memory Start Address*

*--R4: Stores 32 bit S array value*

*--R5: Stores A bar*

*--R6: Stores B bar*

*--R7: Rotate amt*

*--R8: Rotate loop counter*

*--R9: Temp A*

*--R10: Temp B*

*--R11: LOOP COUNTER MAX = 12*

## **Rotate Logic:**

Since the instruction set contains only shift instructions, an alternate logic needs to be used to implement data dependent rotates.

Consider a