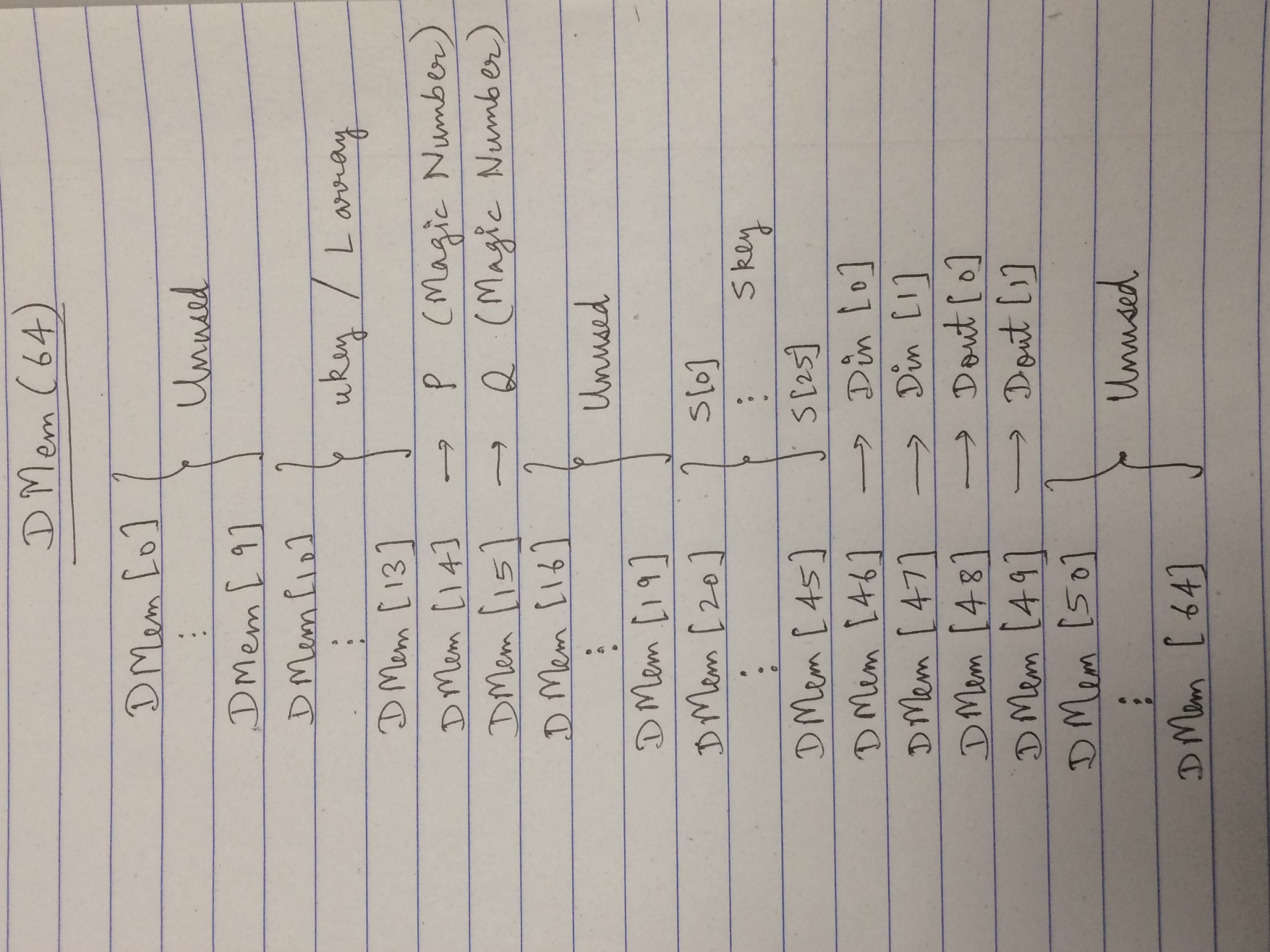
RC5 Assembly Codes

# Data Memory Distribution

****

# // Key Generation

ADDI R0 R1 0xB7E1 // Instructions to initialize P and Q magic Constants

SHL R1 R1 0x0010

ORI R1 R1 0x5163

SW R0 R1 0x000E // Storing Pw

ADDI R00 R02 0x9E37

SHL R02 R02 0x0010

ORI R02 R02 0x79B9

SW R00 R02 0x000F // Storing Qw

// Initializing S array:

// P = R1, Q = R2, R6 = (Length of S array - 1), R4 = Dmem Location for S array

Lw R0, R1, 14 // R1 = P

Lw R0, R2, 15 // R2 = Q

Addi R0, R4, 20 // R4 = 20

Sw R4, R1, 0 // **S[0] = P**

Addi R0, R6, 25 // Initializing R6 = 25

Add R4, R6, R7 // R7 = 46 // End location of S array

LoopKeyInit:

Addi R4, R4, 1 // increment i

Lw R4, R8, 0xFFFF // S[ i - 1 ]

add R8, R2, R9 // R9 = R2 + R8 // S[i] = S[i-1] + Qw

Sw R4, R9, 0 // **S[i] = S[i-1] + Q**

BLT R4, R7, LoopKeyInit // loop through for 25 times.

// Update S and L (Key expansion):

// A=R1, B=R2, i = R3, j = R4, k = R5

Addi R0, R1, 0 // Initialize A = 0

Addi R0, R2, 0 // Initialize B = 0

Addi R0, R4, 0 // Initialize j = 0

Addi R0, R5, 0 // Initialize k = 0

Addi R0, R6, 3 // Outer loop count

Addi R0, R7, 26 // Inner loop count

LoopKeyExp1:

Addi R0, R3, 0 //Initialize i = 0

LoopKeyExp2:

// Update S array begins

Add R1, R2, R8 // A+B

Lw R3, R9, 20 // S[i]

Add R8, R9, R10 // S[i] + (A+B)

SHL R10, R11, 3 // R11 = R10 << 3

SHR R10, R12, 29 // R12 = R10 >> 29

OR R11, R12, R1 // A = R11 OR R12 // A = R9 <<< 3

Sw R3, R1, 20 // **S[i] = A**

// Update L array begins

Add R1, R2, R8 // A+B

Lw R4, R9, 10 // L[j]

Add R8, R9, R10 // L[j] + (A+B)

Addi R10, R13, 0 // copy of R10

ANDI R8, R8, 0x0000001F //mod 32 of (A+B) //Left Shift Amount

Addi R0, R11, 32

Sub R11, R8, R11 // R11= 32-R8 // Right Shift Amount

// Data Dependent Circular Left Rotate begins

Addi R0, R12, 0 // Initializing R12

LoopSHL:

BEQ R12, R8, EndLoopSHL

SHL R10, R10, 1

Addi R12, R12, 1 // Incrementing R12

Jmp LoopSHL

EndLoopSHL:

Addi R0, R12, 0 // Initializing R12

LoopSHR:

BEQ R12, R11, EndLoopSHR

SHR R13, R13, 1

Addi R12, R12, 1 // Incrementing R12

Jmp LoopSHR

EndLoopSHR:

OR R10, R13, R2 // B = R10 OR R13 // Result of Data Dependent Circular Left Rotate

Sw R4, R2, 10 // **L[j] = B**

Addi R4, R4, 1 //Increment j

ANDI R4, R4, 0x00000003 //Mod 4 of j

Addi R3, R3,1 // Increment i

BLT R3, R7, LoopKeyExp2

Addi R5, R5, 1 // Increment k

BLT R5, R6, LoopKeyExp1

# // RC5 Encryption

Lw R0, R1, 46 // Initialize R1: A = Din [0]

Lw R0, R2, 47 // Initialize R2: B = Din [1]

Lw R0, R17, 20 // S[0]

Lw R0, R18, 21 // S[1]

Add R1, R17, R1 // Pre-round A

Add R2, R18, R2 // Pre-round B

Addi R0, R3, 1 // Initialize i = 1

Addi R0, R16, 13 // Loop Count

LoopEnc:

// Computation for A begins

// XOR begins

NOR R1, R1, R4 // R4 = NOT A

NOR R2, R2, R5 // R5 = NOT B

NOR R1, R2, R6 // R6 = A NOR B

NOR R4, R5, R7 // R7 = (NOT A) NOR (NOT B)

NOR R6, R7, R8 // R8 = A XOR B

Add R0, R8, R9 // copy of R8

ANDI R2, R10, 0x0000001F //mod 32 of B //Left Shift Amount

Addi R0, R11, 32

Sub R11, R10, R11 // R11= 32-R10 // Right Shift Amount

// Data Dependent Circular Left Rotate begins

Addi R0, R12, 0 // initializing R12 = 0

LoopSHL2:

BEQ R12, R10, EndLoopSHL2

SHL R8, R8, 1

Addi R12, R12, 1 // Incrementing R12

Jmp LoopSHL2

EndLoopSHL2:

Addi R0, R12, 0 // initializing R12 = 0

LoopSHR2:

BEQ R12, R11, EndLoopSHR2

SHR R9, R9, 1

Addi R12, R12, 1 // Incrementing R12

Jmp LoopSHR2

EndLoopSHR2:

OR R8, R9, R13 // R13= R8 OR R9

// Result of Data Dependent Circular Left Rotate ready

Add R3, R3, R14

Lw R14, R15, 20 // S [ 2\*i ]

Add R13, R15, R1 // A = Rotated value + S value

// Computation for B begins

// XOR begins

NOR R1, R1, R4 // R4 = NOT A

NOR R2, R2, R5 // R5 = NOT B

NOR R2, R1, R6 // R6 = B NOR A

NOR R5, R4, R7 // R7 = (NOT B) NOR (NOT A)

NOR R7, R6, R8 // R8 = B XOR A

Add R0, R8, R9 // copy of R8

ANDI R1, R10, 0x0000001F //mod 32 of A //Left Shift Amount

Addi R0, R11, 32

Sub R11, R10, R11 // R11= 32-R10 // Right Shift Amount

// Data Dependent Circular Left Rotate begins

Addi R0, R12, 0 // initializing R12 = 0

LoopSHL3:

BEQ R12, R10, EndLoopSHL3

SHL R8, R8, 1

Addi R12, R12, 1 // Incrementing R12

Jmp LoopSHL3

EndLoopSHL3:

Addi R0, R12, 0 // initializing R12 = 0

LoopSHR3:

BEQ R12, R11, EndLoopSHR3

SHR R9, R9, 1

Addi R12, R12, 1 // Incrementing R12

Jmp LoopSHR3

EndLoopSHR3:

OR R8, R9, R13 // R13= R8 OR R9

// Result of Data Dependent Circular Left Rotate ready

Add R3, R3, R14

Lw R14, R15, 21 // S [ 2\*i + 1]

Add R13, R15, R2 // B = Rotated value + S value

Addi R3, R3, 1 // Increment i

BLT R3, R16, LoopEnc // Loop condition check

Sw R0, R1, 48 // Dout [0] 🡨 A

Sw R0, R2, 49 // Dout [1] 🡨 B

# // RC5 Decryption

Lw R0, R1, 46 // Initialize R1: A = Din [0]

Lw R0, R2, 47 // Initialize R2: B = Din [1]

Addi R0, R3, 12 // Initialize i = 12

LoopDec:

// Computation for B begins

Add R3, R3, R14

Lw R14, R15, 21 // S [ 2\*i + 1]

Sub R2, R15, R8 // R8 = B - S value

Add R0, R8, R9 // copy of R8

ANDI R1, R10, 0x0000001F //mod 32 of A //Right Shift Amount

Addi R0, R11, 32

Sub R11, R10, R11 // R11= 32-R10 // Left Shift Amount

// Data Dependent Circular Right Rotate begins

Addi R0, R12, 0 // initializing R12 = 0

LoopSHR4:

BEQ R12, R10, EndLoopSHR4

SHR R8, R8, 1

Addi R12, R12, 1 // Incrementing R12

Jmp LoopSHR4

EndLoopSHR4:

Addi R0, R12, 0 // initializing R12 = 0

LoopSHL4:

BEQ R12, R11, EndLoopSHL4

SHL R9, R9, 1

Addi R12, R12, 1 // Incrementing R12

Jmp LoopSHL4

EndLoopSHL4:

OR R8, R9, R13 // R13= R8 OR R9

// Result of Data Dependent Circular Right Rotate ready

// XOR begins

NOR R1, R1, R4 // R4 = NOT A

NOR R13, R13, R5 // R5 = NOT R13

NOR R13, R1, R6 // R6 = R13 NOR A

NOR R5, R4, R7 // R7 = (NOT R13) NOR (NOT A)

NOR R7, R6, R2 // B: R2 = R13 XOR A

// Computation for A begins

Add R3, R3, R14

Lw R14, R15, 20 // S [ 2\*i ]

Sub R1, R15, R8 // R8 = A - S value

Add R0, R8, R9 // copy of R8

ANDI R2, R10, 0x0000001F //mod 32 of B //Right Shift Amount

Addi R0, R11, 32

Sub R11, R10, R11 // R11= 32-R10 // Left Shift Amount

// Data Dependent Circular Right Rotate begins

Addi R0, R12, 0 // initializing R12 = 0

LoopSHR5:

BEQ R12, R10, EndLoopSHR5

SHR R8, R8, 1

Addi R12, R12, 1 // Incrementing R12

Jmp LoopSHR5

EndLoopSHR5:

Addi R0, R12, 0 // initializing R12 = 0

LoopSHL5:

BEQ R12, R11, EndLoopSHL5

SHL R9, R9, 1

Addi R12, R12, 1 // Incrementing R12

Jmp LoopSHL5

EndLoopSHL5:

OR R8, R9, R13 // R13= R8 OR R9

// Result of Data Dependent Circular Right Rotate ready

// XOR begins

NOR R13, R13, R4 // R4 = NOT R13

NOR R2, R2, R5 // R5 = NOT B

NOR R13, R2, R6 // R6 = R13 NOR B

NOR R4, R5, R7 // R7 = (NOT R13) NOR (NOT B)

NOR R6, R7, R1 // A: R1 = R13 XOR B

Subi R3, R3, 1 // Decrement i

BNE R3, R0, LoopDec // Loop condition check

Lw R0, R17, 20 // S[0]

Lw R0, R18, 21 // S[1]

Sub R1, R17, R1 // Post-round A

Sub R2, R18, R2 // Post-round B

Sw R0, R1, 48 // Dout [0] 🡨 A

Sw R0, R2, 49 // Dout [1] 🡨 B