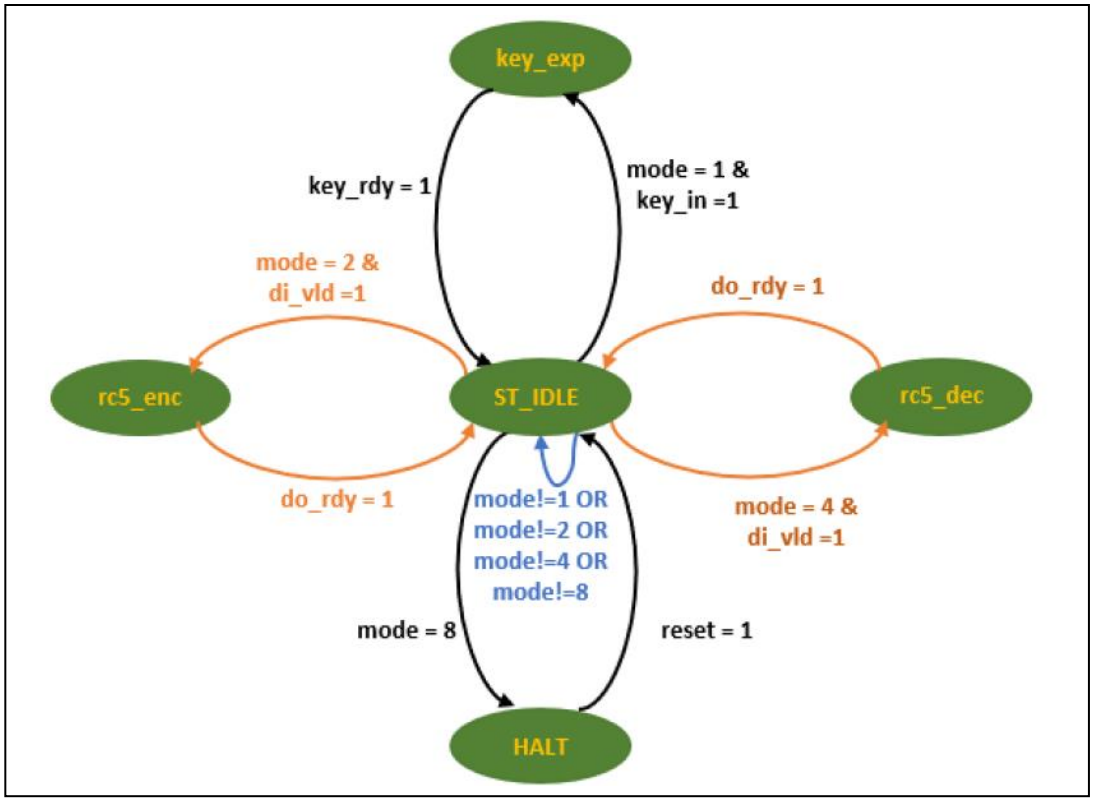
State Diagram



Description of RC5 Implementation in Assembly

RC5 implementation in Assembly consists mainly of 3 processes: Key Expansion, Encryption, and Decryption.

1. Key Expansion: Key expansion involves initialization of L-array, S-array and mixing the keys to form S array.

   In the initialization part, we stored magic constants P(Mem [128]) and Q(Mem [132]) in the data memory and then fetched them to register file to perform key expansion. We created few counters to execute loop instructions. The MSB 96 bits of user key is hardcoded to zeros by copying R0 content to specific memory locations where user key is stored. The LSB 32 bits of user key is loaded from memory. The S[0] and S[1] are initialized to P and P+Q respectively.

$$\begin{aligned}&\textbf{for } i = b-1 \ \textbf{ downto } \ 0 \ \textbf{do}\\ &\qquad L[i/u] = (L[i/u] \lll 8) + K[i];\end{aligned}$$

The L array is created by shifting user key and storing it in L array. L[3] starts from mem[112] to mem[115] and ends at L[0], mem[124] to mem[127].

$$S[0] = P_w;$$
$$\textbf{for } i = 1 \textbf{ to } t-1 \textbf{ do}$$
$$\qquad S[i] = S[i-1] + Q_w;$$

The S array ranges from S[0] to S[25]. S[0] is stored in the data memory starting from mem[8] to mem[11] storing 32-bits at four memory locations 8-bit each. Similarly, other S array elements were stored in mem[108] to mem[111]. It is always computed as per the RC5 algorithm when the key expansion is performed.

$$i = j = 0;$$
$$A = B = 0;$$
$$\textbf{do } 3 * \max(t, c) \textbf{ times:}$$
$$\qquad A = S[i] = (S[i] + A + B) \lll 3;$$
$$\qquad B = L[j] = (L[j] + A + B) \lll (A + B);$$
$$\qquad i = (i + 1) \bmod(t);$$
$$\qquad j = (j + 1) \bmod(c);$$

In the last step, S and L array mixing is performed by shifting operations. Data independent left shift is performed on S array and data dependent left shifting is performed on L array.

2. Encryption

The encryption had data dependent left rotate. It is total 12 time shift, so counter is stored in R16 register. R19 register is always 1. At first, A + S[0] is stored in R3, B + S[1] is stored in R6.

```
void RC5_ENCRYPT(WORD *pt, WORD *ct) /* 2 WORD input pt/output ct    */
{ WORD i, A=pt[0]+S[0], B=pt[1]+S[1];
  for (i=1; i<=r; i++)
    { A = ROTL(A^B,B)+S[2*i];
      B = ROTL(B^A,A)+S[2*i+1];
    }
  ct[0] = A; ct[1] = B;
}
```

Then we performed A^B, stored in R4. It is left rotated by B and stored in R10. Final rotation is added with S[2*i] and stored in R1. B^A is stored in R4 and rotated value is stored in R15 and added with S[2*i + 1] is stored in R2 register.
So, we get encrypted value in R1 and R2 register.

3. Decryption:

The decryption had data dependent right rotate. It is a total 12-time shift, so the counter is stored in a register.

```
void RC5_DECRYPT(WORD *ct, WORD *pt) /* 2 WORD input ct/output pt    */
{ WORD i, B=ct[1], A=ct[0];
  for (i=r; i>0; i--)
    { B = ROTR(B-S[2*i+1],A)^A;
      A = ROTR(A-S[2*i],B)^B;
    }
  pt[1] = B-S[1]; pt[0] = A-S[0];
}
```

B - S[2*i + 1] is stored in R7 register. A - S[2*i] is stored in R8. S[0] and S[1] were stored in R3 and R4 respectively. So after performing loop operation, B - S[1] was stored in R2 and A - S[0] was stored in R1. The final decrypted value was stored in R1 and R2 register.