PROYECTO NO.3
ALGORITMO DE BOOTH PROCESADOR PDUA

DAVID JULIÁN CUADROS ASTRO
ERICK SANTIAGO CAMARGO GARCÍA
ESTUDIANTES INGENIERÍA DE SISTEMAS

CRISTIAN DÍAZ ÁLVAREZ
DOCENTE

PONTIFICIA UNIVERSIDAD JAVERIANA
ARQUITECTURA DEL COMPUTADOR
FACULTAD DE INGENIERÍA
BOGOTÁ D.C.
2023

Para el proyecto No.3 se realizó el algoritmo de Booth que consiste en la multiplicación por software para el procesador PDUA con sus listas de instrucciones. A continuación, se presenta el desarrollo del código:

Ejercicio: Algoritmo de Booth

```
#PASAMOS VALOR AL CONTADOR
        MOVE ACC, CTE
        0
        MOVE DPTR, ACC
        MOVE ACC, [DPTR]
        MOVE CONT, ACC

#PASAMOS VALOR A A (ACUMULADOR)
        MOVE ACC, CTE
        00000000
        MOVE DPTR, ACC
        MOVE ACC, [DPTR]
        MOVE A, ACC
-------------------------------------------------------
#PASAMOS VALOR DEL MULTIPLICADOR Q
        MOVE ACC, CTE
        multiplicador
        MOVE DPTR, ACC
        MOVE ACC, [DPTR]
        MOVE Q, ACC
-------------------------------------------------------
#GUARDAMOS VALOR DE Q-1
        MOVE ACC, CTE
        0
        MOVE DPTR, ACC
        MOVE ACC, [DPTR]
        MOVE Q-1, ACC //Inicialmente se le da el valor de 0
-------------------------------------------------------------------------
#PASAMOS VALOR DEL MULTIPLICANDO M
        MOVE ACC, CTE
        multiplicando
        MOVE DPTR, ACC
        MOVE ACC, [DPTR]
        MOVE M, ACC

COMPARACIÓN:
-------------------------------------------------------
#GUARDAMOS VALOR DE Q0
        MOVE Q0, Q
        AND Q0, 00000001 //Guardamos en Q0 el bit menos significativo
-------------------------------------------------------------------------
#RESTAMOS PARA HACER LOS SALTOS SEGÚN EL CASO
        MOVE VAR1, Q0
        INV VAR1
        ADD VAR1, 1 #Complemento A2
        INV VAR1 #Resta
```

```
        MOVE VAR2, Q-1
        INV VAR2
        ADD VAR2, 1 #Complemento A2
        INV VAR2 #Resta 1

#SALTOS
        JZ VAR1 #Si var1 es 0 puede haber la opción de que Q0, Q-1 sean (10-11)

CONDICION_Q0=
        JMP CONDICION_Q-1=0

CONDICION_Q0=1:
        JZ VAR2 #Si var2 es 0 es porque Q0, Q-1 son 11
        RUTA_11_00; #La misma ruta para las dos opciones (11,00)
        JMP
        RUTA_10

CONDICION_Q-1=0
        JZ VAR2 #Si var2 es 0 es porque Q0, Q-1 son 01
        RUTA_01
        JMP #Si var2 no es 0 es porque Q0, Q-1 son 00
        RUTA_11_00
RUTA_10:
    MOVE AUXM, M #Guardar el valor de M en otra variable para no perder su valor
    original
    INV AUXM
    ADD AUXM, 1
    ADD A, AUXM #Hacemos la resta A-M
    JMP CORRIMIENTO

RUTA_01:
    ADD A, M #Sumar A y M y guardamos en A
    JMP CORRIMIENTO

RUTA_11_00:
    JMP CORRIMIENTO

CORRIMIENTO=
    MOVE MSB_A, A
    AND MSB_A, 10000000 #Guardamos el bit más significativo de A (Acumulador)
    MOVE LSB_A, A
    AND LSB_A, 00000001 #Guardamos el bit menos significativo de A

    MOVE LSB_Q, Q
    AND LSB_Q, 00000001 #Guardamos el bit menos significativo de Q


    MOVE Q-1, LSB_Q #Actualizamos el valor de Q-1

    SHR A #Corrimiento de A (Acumulador)
    SHR Q #Corrimiento de Q
```

```
#Hacemos resta para los saltos
INV MSB_A
ADD MSB_A, 1
INV MSB_A #Restamos al más significativo del acumulador 1 para ver si era 1 o 0


INV LSB_A
ADD LSB_A, 1
INV LSB_A #Restamos al menos significativo del acumulador 1 para ver si era 1 o 0


INV LSB_Q
ADD LSB_Q, 1
INV LSB_Q #Restamos al menos significativo de Q 1 para ver si da 0 o 1

JZ MSB_A
        ADD A, 10000000 #Para que el más significativo de A sea 1, si no entra a esta
        opción, al hacer el corrimiento, el bit más significativo será 0.

JZ LSB_A
        ADD Q, 10000000 #Para que el más significativo de Q sea 1, si no entra a esta
        opción, al hacer el corrimiento, el bit más significativo de Q será 0

JZ LSB_Q
        MOVE Q-1, 1 #Guardamos en Q-1 el valor de 1

ADD CONT, 1 #Sumar 1 al contador

#Concatenar todos los resultados para dar el resultado de la multiplicación

MOVE AUXCONT, CONT
ADD AUXCONT, 11000
#Operación (Auxcont +(-8)) para saber si terminamos el algoritmo

JZ AUXCONT
        AND A, 11111111000000000 #Para agregar ceros a la derecha
        AND Q, 111111110 #Para agregar ceros a la derecha
        ADD Q, A
        ADD Q, Q-1; #Sumamos todos los valores finales de cada variable (A, Q, Q-1)
        MOVE RTA, Q #Resultado final
        JMP FIN

JMP COMPARACION
```

FIN

Multiplicador: 10000101
Multiplicando: 11001101

Pontificia Universidad JAVERIANA Bogotá

Ejercicio: Potenciación

```
#PASAMOS VALOR AL CONTADOR
        MOVE ACC, CTE
        0
        MOVE DPTR, ACC
        MOVE ACC, [DPTR]
        MOVE CONT, ACC

#PASAMOS VALOR Y
        MOVE ACC, CTE
        Y
        MOVE DPTR, ACC
        MOVE ACC, [DPTR]
        MOVE Y, ACC
        INV Y
        ADD Y, 1
        INV Y #Restamos 1 para que no haya un ciclo más

#PASAMOS VALOR A A (ACUMULADOR)
        MOVE ACC, CTE
        00000000
        MOVE DPTR, ACC
        MOVE ACC, [DPTR]
        MOVE A, ACC
-------------------------------------------------------
#PASAMOS VALOR DEL MULTIPLICADOR Q
        MOVE ACC, CTE
        X
        MOVE DPTR, ACC
        MOVE ACC, [DPTR]
        MOVE Q, ACC
        MOVE M, ACC
-------------------------------------------------------
#GUARDAMOS VALOR DE Q-1
        MOVE ACC, CTE
        0
        MOVE DPTR, ACC
        MOVE ACC, [DPTR]
        MOVE Q-1, ACC //Inicialmente se le da el valor en 0

COMPARACIÓN:
-------------------------------------------------------
#GUARDAMOS VALOR DE Q0
        MOVE Q0, Q
        AND Q0, 00000001 //Guardamos en Q0 el bit menos significativo
-------------------------------------------------------
#RESTAMOS PARA HACER LOS SALTOS SEGÚN EL CASO
        MOVE VAR1, Q0
```

```
        INV VAR1
        ADD VAR1, 1 #Complemento A2
        INV VAR1 #Resta 1

        MOVE VAR2, Q-1
        INV VAR2
        ADD VAR2, 1 #Complemento A2
        INV VAR2 #Resta 1

#SALTOS
        JZ VAR1 #Si var1 es 0 puede haber la opción de que Q0, Q-1 sean (10-11)

CONDICION_Q0=1
        JMP CONDICION_Q-1=0

CONDICION_Q0=1:
        JZ VAR2 #Si var2 es 0 es porque Q0, Q-1 son 11
        RUTA_11_00; #La misma ruta para las 2 opciones (11,00)
        JMP
        RUTA_10

CONDICION_Q-1=0
        JZ VAR2 #Si var2 es 0 es porque Q0, Q-1 son 01
        RUTA_01
        JMP #Si var2 no es 0 es porque Q0, Q-1 son 00
        RUTA_11_00

RUTA_10:
        MOVE AUXM, M #Guardar el valor de M en otra variable para perder su valor
        original
        INV AUXM
        ADD AUXM, 1
        ADD A, AUXM #Hacemos la resta de A-M
        JMP CORRIMIENTO

RUTA_01:
        ADD A, M #Sumar A y M y guardamos en A
        JMP CORRIMIENTO

RUTA_11_00:
        JMP CORRIMIENTO

CORRIMIENTO=
        MOVE MSB_A, A
        AND MSB_A, 10000000 #Guardamos el bit más significativo de A (Acumulador)
        MOVE LSB_A, A
        AND LSB_A, 00000001 #Guardamos el bit menos significativo de A

        MOVE LSB_Q, Q
        AND LSB_Q, 00000001 #Guardamos el bit menos significativo de Q

        MOVE Q-1, LSB_Q #Actualizamos el valor de Q-1

        SHR A #Corrimiento de A (Acumulador)
        SHR Q #Corrimiento de Q
```

```
#Hacemos resta para los saltos
INV MSB_A
ADD MSB_A, 1
INV MSB_A #Restamos al más significativo del acumulador 1 para ver si era 1 o 0

INV LSB_A
ADD LSB_A, 1
INV LSB_A #Restamos al menos significativo del acumulador 1 para ver si era 1 o 0


INV LSB_Q
ADD LSB_Q, 1
INV LSB_Q #Restamos al menos significativo de Q 1 para ver si da 0 o 1

JZ MSB_A
        ADD A, 10000000 #Para que el más significativo de A sea 1, si no entra a esta
        opción, al hacer el corrimiento, el bit más significativo será 0

JZ LSB_A
        ADD Q, 10000000 #Para que el más significativo de Q sea 1, sino entra a esta
        opción, al hacer el corrimiento, el bit más significativo de Q será 0

JZ LSB_Q
        MOVE Q-1, 1 #Guardamos en Q-1 el valor de 1

ADD CONT, 1 #Sumar 1 al contador
MOVE AUXCONT, CONT
ADD AUXCONT, Y # (Y ya es negativo, se hizo al inicio) Operación para saber si
terminamos el algoritmo

JZ AUXCONT

#Concatenar todos los resultados para dar el resultado de la multiplicación
AND A, 1111111000000000 #Para agregar ceros a la derecha
AND Q, 111111110 #Para agregar ceros a la derecha
ADD Q, A
ADD Q, Q-1; #Sumamos todos los valores finales de cada variable (A, Q, Q-1)

JMP COMPARACION
```

```
FIN

X: 10000101
Y: 11001101
```