Computer Structure and Language

Hamid Sarbazi-Azad
Department of Computer Engineering
Sharif University of Technology (SUT)
Tehran, Iran

RS Instructions:

OPCODE	r1	r3	B2		D2

Load Multiple

Mnemonic: LM r1,r3,S2
LM r1,r3,D2(B2)
LM r1,r3,D2

Operation: r1 ← $(M_{D2+(B2)})$;
r1+1 ← $(M_{D2+(B2)+4})$;
…
r3 ← $(M_{D2+(B2)+(r3-r1)*4})$;

OPCODE: 98h

Note 1: Storage address S2 = D2 + (B2) must be a multiple of 4 (word aligned).
Note 2: if r1>r3, first register r۱-R15 will be loaded from the address provided and then registers R0-r۳ will be loaded thereafter.

RS Instructions:

OPCODE	r1	r3	B2	D2

Load Multiple

Example 1:

Assembly instruction: LM 2,6,ARRAY

Symbol ARRAY has an address formed by base register 12 and displacement value 124h.

Operation:

$R2 \leftarrow (M_{(R12)+124h});$
$R3 \leftarrow (M_{(R12)+128h});$
$R4 \leftarrow (M_{(R12)+12Ch});$
$R5 \leftarrow (M_{(R12)+130h});$
$R6 \leftarrow (M_{(R12)+134h});$

Machine code: 9826C124

RS Instructions:

OPCODE	r1	r3	B2	D2

Load Multiple

Example 2:

Assembly instruction: LM 13,3,6(5)

Operation:

$R13 \leftarrow (M_{(R5)+6});$
$R14 \leftarrow (M_{(R5)+10});$
$R15 \leftarrow (M_{(R5)+14});$
$R0 \leftarrow (M_{(R5)+18});$
$R1 \leftarrow (M_{(R5)+22});$
$R2 \leftarrow (M_{(R5)+26});$
$R3 \leftarrow (M_{(R5)+30});$

Machine code: 98D35006

RS Instructions:

OPCODE	r1	r3	B2		D2

Store Multiple

Mnemonic: STM r1,r3,S2
STM r1,r3,D2(B2)
STM r1,r3,D2

Operation: $M_{D2+(B2)} \leftarrow (r1);$
$M_{D2+(B2)+4} \leftarrow (r1+1);$
…
$M_{D2+(B2)+(r3-r1)*4} \leftarrow (r3);$

OPCODE: 90h

Note 1: Storage address S2 = D2 + (B2) must be a multiple of 4 (word aligned).
Note 2: if r1>r3, first register r۱-R15 are stored in the memory location given and then registers R0-r۳ are stored thereafter.

RS Instructions:

OPCODE	r1	r3	B2		D2

Store Multiple

Example 1:

Assembly instruction: **STM** 2,6,ARRAY — Symbol ARRAY has an address formed by base register 12 and displacement value 124h.

Operation: $M_{(R12)+124h} \leftarrow (R2);$
$M_{(R12)+128h} \leftarrow (R3);$
$M_{(R12)+12Ch} \leftarrow (R4);$
$M_{(R12)+130h} \leftarrow (R5);$
$M_{(R12)+134h} \leftarrow (R6);$

Machine code: 9026C124

RS Instructions:

OPCODE	r1	r3	B2	D2

Store Multiple

Example 2:

Assembly instruction: STM 13,3,6(5)

Operation:

$M_{(R5)+6} \leftarrow (R13);$
$M_{(R5)+10} \leftarrow (R14);$
$M_{(R5)+14} \leftarrow (R15);$
$M_{(R5)+18} \leftarrow (R0);$
$M_{(R5)+22} \leftarrow (R1);$
$M_{(R5)+26} \leftarrow (R2);$
$M_{(R5)+30} \leftarrow (R3);$

Machine code: 90D35006

RS Instructions: Shift Instructions

OPCODE	r1	0	B2	D2

Shift instructions are coded in RS format and use the following generic mnemonic form:

Sxy r1,D2(B2) for single-register shift
SxDy r1,D2(B2) for register-pair shift

- The amount of shift is given by D2+(B2) value.
- x="L": shift **L**eft is performed.
- x="R": shift **R**ight is performed.
- y="L": **L**ogical shift is performed.
- y="A": **A**rithmetic shift is performed.

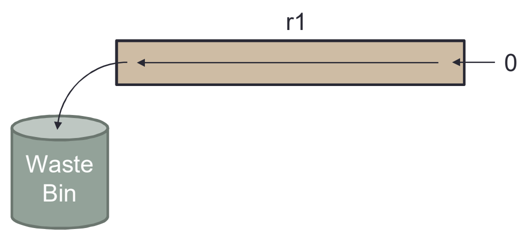
→ We have: SLL, SLA, SRL, SRA for single-register shifts, and SLDL, SLDA, SRDL, SRDA for register-pair shifts.

RS Instructions: Shift Left Logical

OPCODE	r1	0	B2	D2

Mnemonic: SLL r1,D2(B2)
SLL r1,D2

Operation: $r1 \leftarrow (r1) << D2+(B2)_{4\text{-}0}$; Five least-significant bits used.
OPCODE: 89h



RS Instructions: Shift Left Logical

OPCODE	r1	0	B2	D2

Example 1:
Assembly instruction: SLL 2,7 Suppose (R2) = 12345678h
Operation: R2 ← 1A2B3C00h;
Machine code: 89200007

0001 0010 0011 0100 0101 0110 0111 1000
0001 0010 0011 0100 0101 0110 0111 1000 ← 0000000
0001 0010 0011 0100 0101 0110 0111 1000 0000000
1 A 2 B 3 C 0 0

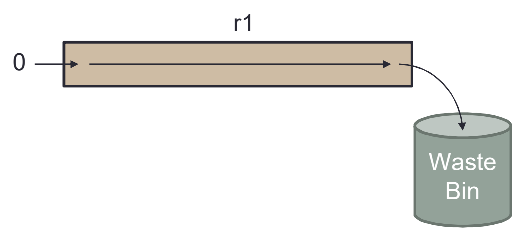
Example 2:
Assembly instruction: SLL 5,2 Suppose (R5) = 500
Operation: R5 ← 2000;
Machine code: 89500002

RS Instructions: Shift Right Logical

OPCODE	r1	0	B2	D2

Mnemonic: SRL r1,D2(B2)
SRL r1,D2

Operation: r1 ← (r1) >> D2+$(B2)_{4\text{-}0}$; Five least-significant bits used.
OPCODE: 88h



RS Instructions: Shift Right Logical

OPCODE	r1	0	B2	D2

Example 1:
Assembly instruction: SRL 2,5 Suppose (R2) = 12345678h
Operation: R2 ← 0091A2B3;
Machine code: **88**200005

0001 0010 0011 0100 0101 0110 0111 1000
00000 → 0001 0010 0011 0100 0101 0110 0111 1000
00000 0001 0010 0011 0100 0101 0110 0111 1000
0 0 9 1 A 2 B 3

Example 2:
Assembly instruction: SRL 2,2 Suppose (R2) = 2000
Operation: R2 ← 500;
Machine code: **88**۲00002

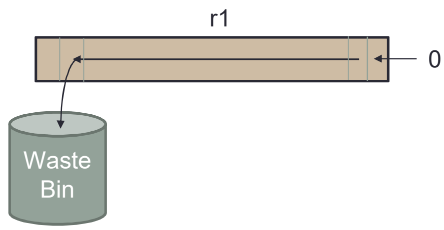
RS Instructions: Shift Left Arithmetic

OPCODE	r1	0	B2	D2

Mnemonic: SLA r1,D2(B2)
SLA r1,D2

Operation: r1 ← (r1) << D2+$(B2)_{4\text{-}0}$; Five least-significant bits used.
OPCODE: 8Bh

Note: Sign bit is not included in shift operation.



RS Instructions: Shift Left Arithmetic

OPCODE	r1	0	B2	D2

Example 1:
Assembly instruction: SLA 2,7 Suppose (R2) = 12345678h
Operation: R2 ← 1A2B3C00h;
Machine code: 8B200007

0001 0010 0011 0100 0101 0110 0111 1000
0001 0010 0011 0100 0101 0110 0111 1000 ← 0000000
0001 0010 0011 0100 0101 0110 0111 1000 0000000
1 A 2 B 3 C 0 0

Example 2:
Assembly instruction: SLA 5,2 Suppose (R5) = -500
Operation: R5 ← -2000;
Machine code: 8B500002

(c) Hamid Sarbazi-Azad — Computer Structure & Language -- Lecture #11: IBM360 Machine — 15

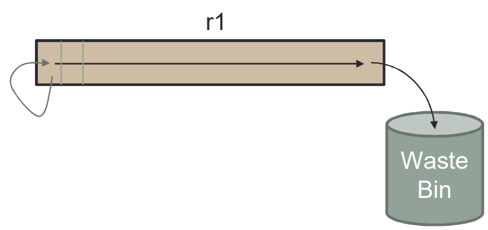
RS Instructions: Shift Right Arithmetic

OPCODE	r1	0	B2	D2

Mnemonic: SRA r1,D2(B2)
SRA r1,D2

Operation: $r1 \leftarrow (r1) >> D2+(B2)_{4\text{-}0}$; Five least-significant bits used.
OPCODE: 8Ah

Note: Sign bit is not included in shift operation.



(c) Hamid Sarbazi-Azad — Computer Structure & Language -- Lecture #11: IBM360 Machine — 16

RS Instructions: Shift Right Arithmetic

OPCODE	r1	0	B2	D2

Example 1:
Assembly instruction: SRA 2,7 — Suppose (R2) = F2345678h
Operation: $R2 \leftarrow$ FFE468ACh;
Machine code: 8A200007

1111 0010 0011 0100 0101 0110 0111 1000
1 1111111→ 111 0010 0011 0100 0101 0110 0111 1000
1 1111111 111 0010 0011 0100 0101 0110 0111 1000
F F E 4 6 8 A C

Example 2:
Assembly instruction: SRA 5,2 — Suppose (R5) = -1000
Operation: $R5 \leftarrow$ -250;
Machine code: 8A500002

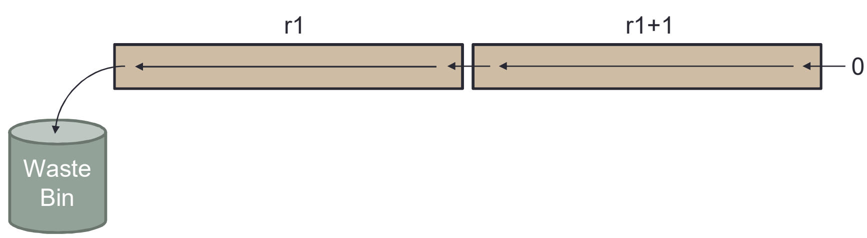
RS Instructions: Shift Left Double Logical

OPCODE	r1	0	B2		D2

Mnemonic: SLDL r1,D2(B2)   r1 must be even
SLDL r1,D2

Operation: r1:r1+1 ← (r1):(r1+1) << D2+$(B2)_{5-0}$;   Six least-significant bits used.

OPCODE: 8Dh



RS Instructions: Shift Left Double Logical

OPCODE	r1	0	B2		D2

Example 1:

Assembly instruction: SLDL 2,7   Suppose (R2) = 12345678h and (R3) = -1

Operation: R2 ← 1A2B3C7Fh;
R3 ← FFFFFF80h;

Machine code: 8D200007

(R2) (R3)

0001 0010 0011 0100 0101 0110 0111 1000 1111 1111 1111 1111 1111 1111 1111 1111

0 0011 0100 0101 0110 0111 1000 1111 1111 1111 1111 1111 1111 1111 1111 ← 0000000

0 0011 0100 0101 0110 0111 1000 1111 1111 1111 1111 1111 1111 1111 1111 0000000

1 A 2 B 3 C 7 F F F F F F F 8 0

Example 2:

Assembly instruction: SLDL 4,32   Suppose (R5) = 500

Operation: R4 ← 500; R5 ← 0;

Machine code: 8D400020

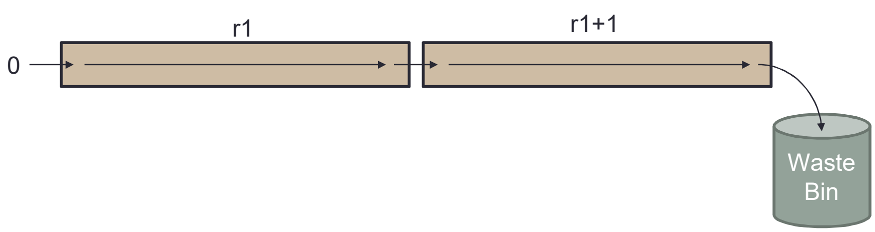
RS Instructions: Shift Right Double Logical

OPCODE	r1	0	B2	D2

Mnemonic: SRDL r1,D2(B2) r1 must be even
SRDL r1,D2

Operation: r1:r1+1 ← (r1):(r1+1) >> D2+$(B2)_{5\text{-}0}$; Six least-significant bits used.

OPCODE: 8Ch



RS Instructions: Shift Right Double Logical

OPCODE	r1	0	B2	D2

Example 1:
Assembly instruction: SRDL 2,4 Suppose (R2) = 12345678h and (R3) = -1
Operation: R2 ← 01۲۳۴۵۶۷h;
R3 ← 8FFFFFFFh;
Machine code: 8C20000۴

(R2) (R3)
0001 0010 0011 0100 0101 0110 0111 1000 1111 1111 1111 1111 1111 1111 1111 1111
0000 → 0001 0010 0011 0100 0101 0110 0111 1000 1111 1111 1111 1111 1111 1111 1111
0 1 2 3 4 5 6 7 8 F F F F F F F

Example 2:
Assembly instruction: SRDL 4,32 Suppose (R4) = 500
Operation: R4 ← 0; R5 ←500;
Machine code: 8C400020

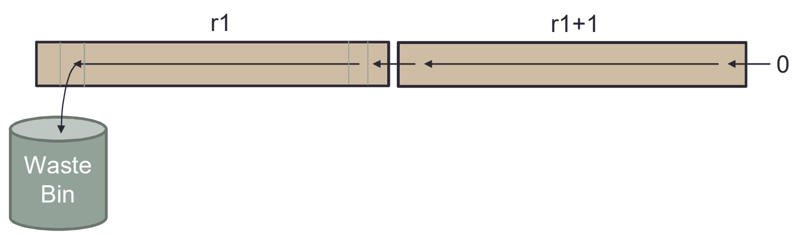
RS Instructions: Shift Left Double Arithmetic

OPCODE	r1	0	B2	D2

Mnemonic: SLDA r1,D2(B2) r1 must be even
SLDA r1,D2

Operation: r1:r1+1 ← (r1):(r1+1) << D2+$(B2)_{5-0}$; Six least-significant bits used.

OPCODE: 8Fh



RS Instructions: Shift Left Double Arithmetic

OPCODE	r1	0	B2	D2

Example 1:

Assembly instruction: SLDA 2,7 Suppose (R2) = C2345678h and (R3) = -1

Operation: R2 ← ٩A2B3C7Fh;
R3 ← FFFFFF80h;

Machine code: 8F200007

(R2) (R3)

1100 0010 0011 0100 0101 0110 0111 1000 1111 1111 1111 1111 1111 1111 1111 1111
1 0011 0100 0101 0110 0111 1000 1111 1111 1111 1111 1111 1111 1111 1111 ← 000 0000
1 0011 0100 0101 0110 0111 1000 1111 1111 1111 1111 1111 1111 1111 1111 000 0000
٩ A 2 B 3 C 7 F F F F F F F 8 0

Example 2:

Assembly instruction: SLDA 4,32 Suppose (R4)=-1000, (R5)=-500

Operation: R4 ← -500; R5 ←0;

Machine code: 8F400020

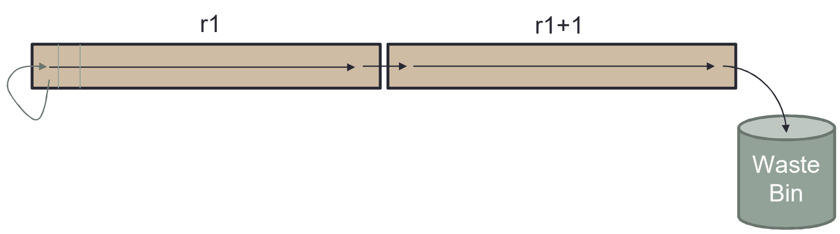
RS Instructions: Shift Right Double Arithmetic

OPCODE	r1	0	B2	D2

Mnemonic: SRDA r1,D2(B2) r1 must be even
SRDA r1,D2

Operation: r1:r1+1 ← (r1):(r1+1) >> D2+$(B2)_{5-0}$; Six least-significant bits used.

OPCODE: 8Eh



RS Instructions: Shift Right Double Arithmetic

OPCODE	r1	0	B2	D2

Example 1:

Assembly instruction: SRDA 2,6 Suppose (R2) = C2345678h and (R3) = -1

Operation: R2 ← FF08D159h;
R3 ← E3FFFFFFh;

Machine code: 8E200006

(R2) (R3)

1100 0010 0011 0100 0101 0110 0111 1000 1111 1111 1111 1111 1111 1111 1111 1111

111111→ 100 0010 0011 0100 0101 0110 0111 1000 1111 1111 1111 1111 1111 1111 11

111111 100 0010 0011 0100 0101 0110 0111 1000 1111 1111 1111 1111 1111 1111 11

F F 0 8 D 1 5 9 E 3 F F F F F F

Example 2:

Assembly instruction: SRDA 4,32 Suppose (R4)=-1000, (R5)=-500

Operation: R4 ← -1; R5 ←-1000;

Machine code: 8E400020

RS Instructions:

OPCODE	r1	r3	B2	D2

Branch on Index Low or Equal

Mnemonic: BXLE r1,r3,S2
BXLE r1,r3,D2(B2)
BXLE r1,r3,D2

Operation: for even r3:
r1 ← (r1)+(r3); if (r1) ≤ (r3+1) then PC ← D2+(B2);
for odd r3:
r1 ← (r1)+(r3); if (r1) ≤ (r3) then PC ← D2+(B2);

OPCODE: 87h

RS Instructions:

OPCODE	r1	r3	B2	D2

Branch on Index Low or Equal

Example 1:
Assembly instruction: BXLE 2,6,LOOP Symbol LOOP has an address formed by base register 12 and displacement value 124h. Suppose (R2) = 4, (R6)=4, and (R7)=20.
Operation: R2← 8; PC ← (R12)+124h;
Machine code: 8726C124

Example 2:
Assembly instruction: BXLE 13,3,6(5) Suppose (R13) = -12, (R3)=4.
Operation: R13 ← -8; PC ← (R5) +6;
Machine code: 87D35006

RS Instructions:

OPCODE	r1	r3	B2	D2

Branch on Index Low or Equal

Sample code: Find the minimum element in the last 10 elements of the half-word array DATA (of 20 elements) and store it in half-word MIN (using BXLE with odd/even r3).

```
        L     3,=F'-14'
        LA    5,2
        LH    6,DATA+38
CMP     CH    6,DATA+34(3)
        BC    4,LESS
        LH    6,DATA+34(3)
LESS    BXLE  3,5,CMP
        STH   6,MIN
        …
DATA    DS    ٢٠H
MIN     DS    H
```

```
        LA    3,22
        LA    4,2
        LA    5,38
        LH    6,DATA+20
CMP     CH    6,DATA(3)
        BC    4,LESS
        LH    6,DATA(3)
LESS    BXLE  3,4,CMP
        STH   6,MIN
        …
DATA    DS    20H
MIN     DS    H
```


RS Instructions:

OPCODE	r1	r3	B2	D2

Branch on Index High

Mnemonic: BXH r1,r3,S2
BXH r1,r3,D2(B2)
BXH r1,r3,D2

Operation: for even r3:
r1 ← (r1)+(r3); if (r1) > (r3+1) then PC ← D2+(B2);
for odd r3:
r1 ← (r1)+(r3); if (r1) > (r3) then PC ← D2+(B2);

OPCODE: 86h

RS Instructions:

OPCODE	r1	r3	B2	D2

Branch on Index High

Example 1:

Assembly instruction: BXH 2,6,LOOP

Symbol LOOP has an address formed by base register 12 and displacement value 124h. Suppose (R2) = 4, (R6)=4, and (R7)=4.

Operation: R2← 8; PC ← (R12)+124h;

Machine code: 8626C124

Example 2:

Assembly instruction: BXH 13,3,6(5)

Suppose (R13) = 12, (R3)=-4.

Operation: R13 ← 8; PC ← (R5) +6;

Machine code: 86D35006

RS Instructions:

OPCODE	r1	r3	B2	D2

Branch on Index High

Sample code: Find the minimum element in the last 10 elements of the half-word array DATA (of 20 elements) and store it in half-word MIN (using BXH with odd/even r3).

```
        LA    3,16
        L     5,=F'-2'
        LH    6,DATA+20
CMP     CH    6,DATA+22(3)
        BC    4,LESS
        LH    6,DATA+22(3)
LESS    BXH   3,5,CMP
        STH   6,MIN
        …
DATA    DS    20H
MIN     DS    H
```

```
        LA    3,38
        L     4,=F'-2'
        LA    5,20
        LH    6,DATA+20
CMP     CH    6,DATA(3)
        BC    4,LESS
        LH    6,DATA(3)
LESS    BXH   3,4,CMP
        STH   6,MIN
        …
DATA    DS    ۲۰H
MIN     DS    H
```

Binary Multiplication

Recall the pen and paper technique we use to multiply n-bit number A and m-bit number B to generate (m+n)-bit number P = A x B.

$$
\begin{array}{rcl}
 & A = a_{n-1} \cdots a_1 a_0 & \times \\
 & B = b_{m-1} \cdots b_1 b_0 & \\
\hline
 & M_0 = a_{n-1}b_0 \; \cdots \; a_1b_0 \; a_0b_0 & \\
 & M_1 = a_{n-1}b_1 \; \cdots \; a_1b_1 \; a_0b_1 & \\
 & \vdots & + \\
 & M_{m-1} = a_{n-1}b_{m-1} \; \cdots \; a_1b_{m-1} \; a_0b_{m-1} & \\
\hline
 & p_{m+n-1} \; p_{m+n-2} \; \cdots \; p_2 \; p_1 \; p_0 &
\end{array}
$$

Binary Multiplication

Multiplication can be done in a serial manner.

Consider n-bit numbers A($a_{n-1}a_{n-2} \dots a_0$) and B($b_{n-1}b_{n-2} \dots b_0$).

We can write:

$$
\begin{aligned}
A \times B &= (A \times 2^{n-1} \times b_{n-1}) + (A \times 2^{n-2} \times b_{n-2}) + \dots + (A \times 2^1 \times b_1) + (A \times b_0) \\
&= [A \ll (n-1)] \times b_{n-1} + [A \ll (n-2)] \times b_{n-2} + \dots + [A \ll 1] \times b_1 + [A \ll 0] \times b_0 \\
&= \Sigma_{i=0..n-1} [A \ll i] \times b_i
\end{aligned}
$$

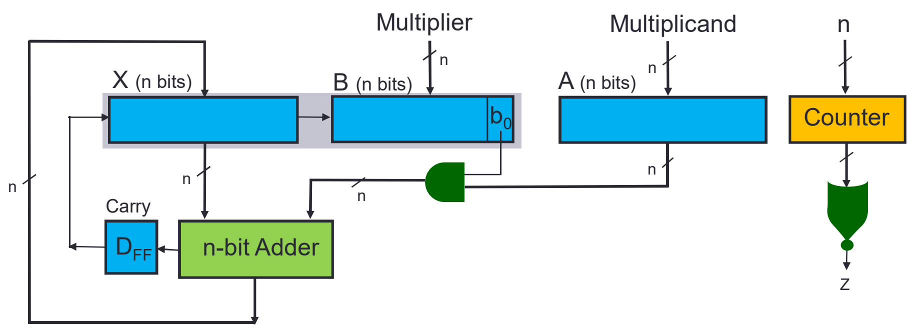
Shifting A for n-1 bits to the left we need a 2n-bit register to keep the partial products.

Alternatively, we can shift the accumulator, that accumulates the partial products, to the right. → a tricky circuit to save hardware ☺

Binary Multiplication

Add and Shift multiplier uses the following circuit (data path):

In the beginning: A and B keep the n-bit multiplicand and multiplier.
At the end: Register pairs X:B contain the 2n-bit multiply result.

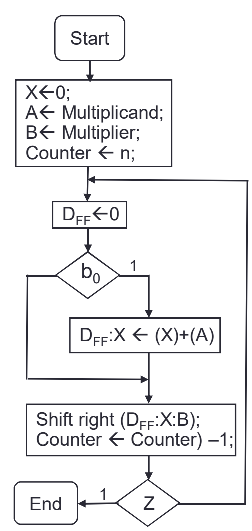


Binary Multiplication

Add & Shift multiplication algorithm is shown.
Example 1: Follow Add & Shift method step by step to multiply 4-bit numbers 9 and 11. (A=1001)

D_{FF}	X	B (b_0)	Counter	Operation
0	0000	1011	4	Add
0	1001	1011	4	Shift
0	0100	1101	3	Add
0	1101	1101	3	Shift
0	0110	1110	2	Shift
0	0011	0111	1	Add
0	1100	0111	1	Shift
0	0110	0011	0	End.

99 = 9 x 11

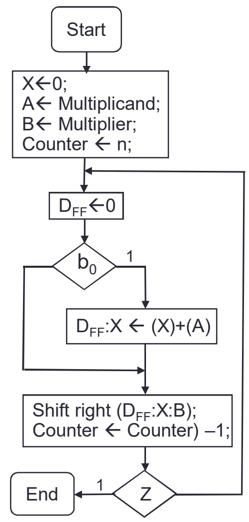


Binary Multiplication

Add & Shift multiplication algorithm is shown.
Example 2: Follow Add & Shift method step by step to multiply 4-bit numbers 12 and 6. (A=1100)

D_{FF}	X	B (b_0)	Counter	Operation
0	0000	0110	4	Shift
0	0000	0011	3	Add
0	1100	0011	3	Shift
0	0110	0001	2	Add
1	0010	0001	2	Shift
0	1001	0000	1	Shift
0	0100	1000	0	End.

72 = 12 x 6



Binary Multiplication

Write a program to multiply (R3) and (R4) and store the result in register pair R2:R3. (i.e. MR 2,4)

```
MULTPLY   START  0
*         Define R12 as base register & initialize it to 6.
*
          XR     2,2      X register
          LA     5,32     Counter
LOOP      LA     6,1
          NR     6,3
          BZ     SHIFT
          AR     2,4
SHIFT     SRDL   2,1
          BCT    5,LOOP

*         Return to OS
          END          MULTIPLY
```

End of Slides