COP800 MathPak

National Semiconductor Application Note 596 Verne H. Wilson June 1989



COP800 MathPak

OVERVIEW

This application note discusses the various arithmetic operations for National Semiconductor's COP800 family of 8-bit microcontrollers. These arithmetic operations include both binary and BCD (Binary Coded Decimal) operation. The four basic arithmetic operations (add, subtract, multiply, divide) are outlined in detail, with several examples shown for both binary and BCD addition and subtraction. Multiplication, division, and BCD conversion algorithms are also provided. Both BCD to binary and binary to BCD conversion subroutines are included, as well as the various multiplication and division subroutines.

Four sets of optimal subroutines are provided for

- 1. Multiplication
- 2. Division
- 3. Decimal (Packed BCD) to binary conversion

4. Binary to decimal (Packed BCD) conversion

One class of subroutines is optimized for minimal COP800 program code, while the second class is optimized for minimal execution time in order to optimize throughput time.

This application note is organized in four different sections. The first section outlines various addition and subtraction routines, including both binary and BCD (Binary Coded Decimal). The second section outlines the multiplication algorithm and provides several optimal multiply subroutines for 1, 2, 3, and 4 byte operation. The third section outlines the division algorithm and provides several optimal division subroutines for 1, 2, 3, and 4 byte operation. The fourth section outlines both the decimal (Packed BCD) to binary and binary to decimal (Packed BCD) conversion algorithms. This section provides several optimal subroutines for these BCD conversions.

The COP800 arithmetic instructions include the Add (ADD), Add with Carry (ADC), Subtract with Carry (SUBC), Increment (INCR), Decrement (DECR), Decimal Correct (DCOR), Clear Accumulator (ACC), Set Carry (SC), and Reset Carry (RC). The shift and rotate instructions, which include the Rotate Right through Carry (RRC) and the Swap Accumulator Nibbles (SWAP), may also be considered as arithmetic instruction variations. The RRC instruction is instrumental in writing a fast multiply routine.

1.0 BINARY AND BCD ADDITION AND SUBTRACTION

In subtraction, a borrow is represented by the absence of a carry and vice versa. Consequently, the carry flag needs to be set (no borrow) before a subtraction, just as the carry flag is reset before an addition. The ADD instruction does not use the carry flag as an input, nor does it change the carry flag. It should also be noted that both the carry and half carry flags (bits 6 and 7, respectively, of the PSW control register) are cleared with reset, and remain unchanged with the ADD, INC, DEC, DCOR, CLR and SWAP instructions. The DCOR instruction uses both the carry and half carry flags. The SC instruction sets both the carry and half carry flags, while the RC instruction resets both these flags. The following program examples illustrate additions and subtractions of 4-byte data fields in both binary and BCD (Binary Coded Decimal). The four bytes from data memory locations 24 through 27 are added to or subtracted from the four bytes in data memory locations 16 through 19. The results replace the data in memory locations 24 through 27. These operations are performed both in Binary and BCD. It should be noted that the BCD pre-conditioning of Adding (ADD) the hex 66 is only necessary with the BCD addition, not with the BCD subtraction. The (Binary Coded Decimal) DCOR (Decimal Correct) instruction uses both the carry and half carry flags as inputs, but does not change the carry and half carry flags. Also note that the #12 with the IFBNE instruction represents 28 - 16, since the IFBNE operand is modulo 16 (remainder when divided by 16).

AN-596

© 1995 National Semiconductor Corporation TL/DD10376

BINARY /	ADDITION:		
	LD	X,#16	: NO LEADING ZERO
	LD	B,#24	; INDICATES DECIMAL
	RC	- , //	; RESET CARRY TO START
L00P:	LD	A,[X+]	; [X] TO ACC
100F :	-		
	ADC	A,[B]	; ADD [B] TO ACC
	Х	A,[B+]	; RESULT TO [B]
	IFBNE	#12	; IF STILL IN DATA FIELD
	JP	LOOP	JUMP BACK TO REPEAT LOOP
	IFC		; IF TERMINAL CARRY,
	JP	OVFLOW	; JUMP TO OVERFLOW
			, JOMI TO OVERTEON
BINARYS	SUBTRACTIO		
	LD	X,#010	; LEADING ZERO
	LD	B,#018	; INDICATES HEX
	SC		; RESET BORROW TO START
L00P:	LD	A,[X+]	; [X] TO ACC
	SUBC	A,[B]	; SUBTRACT [B] FROM ACC
	X	A,[B+]	; RESULT TO [B]
	IFBNE	#12	; IF STILL IN DATA FIELD
	JP	LOOP	; JUMP BACK TO REPEAT LOOP
	IFNC		; IF TERMINAL BORROW,
	JP	NEGRSLT	; JUMP TO NEGATIVE RESULT
BCD ADD			
		W #010	
	LD	X,#010	; LEADING ZERO
	LD	B,#018	; INDICATES HEX
	RC		; RESET CARRY TO START
L00P:	LD	A,[X+]	; [X] TO ACC
	ADD	A,#066	; ADD HEX 66 TO ACC
	ADC	A,[B]	; ADD [B] TO ACC
	DCOR	A	; DECIMAL CORRECT RESULT
	Х	A,[B+]	; RESULT TO [B]
	IFBNE	#12	; IF STILL IN DATA FIELD
	JP	LOOP	; JUMP BACK TO REPEAT LOOP
	IFC		; IF TERMINAL CARRY
	JP	OVFLOW	; JUMP TO OVERFLOW
		0112011	,
BCD SUB	TRACTION:		
	LD	X,#16	; NO LEADING ZERO
	LD	B,#24	; INDICATES DECIMAL
	C		
L00P:	LD	A,[X+]	; [X] TO ACC
2001 •	SUBC	A,[B]	; SUBTRACT [B] FROM ACC
	DCOR	A	; DECIMAL CORRECT RESULT
	Х	A,[B+]	; RESULT TO [B]
	IFBNE	#12	; IF STILL IN DATA FIELD
	JP	LOOP	; JUMP BACK TO REPEAT LOOP
	IFNC		; IF TERMINAL BORROW
	JP	NEGRSLT	
	JL	NEGUSEI	; JUMP TO NEGATIVE RESULT

The astute observer will notice that these previous additions and subtractions are not "adding machine" type arithmetic operations in that the result replaces the second operand rather than the first. The following program examples illustrate "adding machine" type operation where the result replaces the first operand. With subtraction, this entails the result replacing the minuend rather than the subtrahend. Note that the B and X pointers are now reversed.

OVFLOW B,#010 X,018 A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018 A,[X+] A,#066 A,[B]		JUMP TO OVERFLOW B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND RESET BORROW TO START [X] TO ACC EXCHANGE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOP IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
A,[X+] A,[B] A,[B+] #4 LOOP OVFLOW B,#010 X,018 A,[X+] A,[B] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018		RESET CARRY TO START [X] TO ACC ADD [B] TO ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOF IF TERMINAL CARRY JUMP TO OVERFLOW B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND RESET BORROW TO START [X] TO ACC EXCHANGE [B] AND ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOF IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
OVFLOW B,#010 X,018 A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	, . , . , . , . ,	IF TERMINAL CARRY JUMP TO OVERFLOW B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND RESET BORROW TO START [X] TO ACC EXCHANCE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOP IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
OVFLOW B,#010 X,018 A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	, . , . , . , . ,	IF TERMINAL CARRY JUMP TO OVERFLOW B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND RESET BORROW TO START [X] TO ACC EXCHANCE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOP IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
OVFLOW B,#010 X,018 A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	, . , . , . , . ,	IF TERMINAL CARRY JUMP TO OVERFLOW B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND RESET BORROW TO START [X] TO ACC EXCHANCE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOP IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
OVFLOW B,#010 X,018 A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	, . , . , . , . ,	IF TERMINAL CARRY JUMP TO OVERFLOW B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND RESET BORROW TO START [X] TO ACC EXCHANCE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOP IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
OVFLOW B,#010 X,018 A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	, . , . , . , . ,	IF TERMINAL CARRY JUMP TO OVERFLOW B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND RESET BORROW TO START [X] TO ACC EXCHANCE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOF IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
OVFLOW B,#010 X,018 A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	, . , . , . , . ,	IF TERMINAL CARRY JUMP TO OVERFLOW B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND RESET BORROW TO START [X] TO ACC EXCHANCE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOF IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
OVFLOW B,#010 X,018 A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	•, •, •, •, •, •, •, •, •, •, •, •,	IF TERMINAL CARRY JUMP TO OVERFLOW B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND RESET BORROW TO START [X] TO ACC EXCHANGE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOF IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
B,#010 X,018 A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	•, •, •, •, •, •, •, •, •, •, •,	JUMP TO OVERFLOW B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND RESET BORROW TO START [X] TO ACC EXCHANGE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOF IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	•, •, •, •, •, •, •, •, •, •,	X POINTER AT SECOND OPERANI RESET BORROW TO START [X] TO ACC EXCHANGE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOF IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERANI
A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	•, •, •, •, •, •, •, •, •,	X POINTER AT SECOND OPERAND RESET BORROW TO START [X] TO ACC EXCHANGE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOF IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	•, •, •, •, •, •, •, •, •,	X POINTER AT SECOND OPERAND RESET BORROW TO START [X] TO ACC EXCHANGE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOF IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
A,[X+] A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	• • • • • • • • • • • • • • • • • • • •	RESET BORROW TO START [X] TO ACC EXCHANCE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOF IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	• • • • • • • • • • • • • • • • • • • •	<pre>[X] TO ACC EXCHANGE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOF IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND</pre>
A,[B] A,[B] A,[B+] #4 LOOP NEGRSLT B,#010 X,#018	• • • • • • • • • • • • • • • • • • • •	EXCHANGE [B] AND ACC SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOF IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
#4 LOOP NEGRSLT B,#010 X,#018	• • • • • • • • • • • • • • • • • • • •	SUBTRACT [B] FROM ACC RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOM IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
#4 LOOP NEGRSLT B,#010 X,#018	; ; ; ; ;	RESULT TO [B] IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOM IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
#4 LOOP NEGRSLT B,#010 X,#018	; ; ; ;	IF STILL IN DATA FIELD JUMP BACK TO REPEAT LOOH IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERANI
#4 LOOP NEGRSLT B,#010 X,#018	;;;;	JUMP BACK TO REPEAT LOOF IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
NEGRSLT B,#010 X,#018	;;;	IF TERMINAL BORROW JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERAND
B,#010 X,#018	;;;;	JUMP TO NEGATIVE RESULT B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERANI
B,#010 X,#018	;;	B POINTER AT FIRST OPERAND X POINTER AT SECOND OPERANI
X,#018	;	X POINTER AT SECOND OPERAND
X,#018	;	X POINTER AT SECOND OPERAND
X,#018 A,[X+] A,#066 A,[B] A	;	X POINTER AT SECOND OPERAND
A,[X+] A,#066 A,[B] A		
A,[X+] A,#066 A,[B] A	;;;	[X] TO ACC ADD HEX66 TO ACC
A,#066 A,[B] A	;;	ADD HEX66 TO ACC
A,[B] A	;	
A	,	ADD [B] TO ACC
11	•	DECIMAL CORRECT RESULT
A,[B+]		RESULT TO [B]
#4	;	IF STILL IN DATA FIELD
#4 L00P	;	JUMP BACK TO REPEAT LOOP
LOOF	,	IF TERMINAL CARRY
; OVFLOW	;	JUMP TO OVERFLOW
OVI LOW	,	JOMI TO OVERFEON
B,#16	;	
X,#24	;	X POINTER AT SECOND OPERAND
		RESET BORROW TO START
A,[X+]	;	[X] TO ACC
А,[В]	;	EXCHANGE [B] AND ACC
A,[B]	;	SUBTRACT [B] FROM ACC
	;	DECIMAL CORRECT RESULT
A,[B+]	;	RESULT TO [B]
11 4	;	IF STILL IN DATA FIELD
LOOP	;	JUMP BACK TO REPEAT LOOP
	:	IF TERMINAL BORROW
NEGRSLT	;	JUMP TO NEGATIVE RESULT
	A,[B] A A,[B+]	A,[B] ; A ; A,[B+] ;

wish to ac program n	ld five succe nemory to a t	hybrid arithmetic exan ssive bytes of a data wo byte sum, and the byte total TOT. Let us	a table in ROM en subtract the	that the ROM table is located starting at program memoraddress 0401, while SUM and TOT are at RAM data memory locations [1, 0] and [3, 2] respectively, and that we witto encode the program as a subroutine.
ROM Tabl . = 040 . Byte 1 . Byte 4 . Byte 3)1 02 1			
. Byte 2	6			
. Byte 5		- -		
SUMLO SUMLO SUMHI TOTLO TOTHI	= 1 = 2	l op Down		
ARITH1:	LD	X,#5	; SET U	JP ROM TABLE POINTER
	LD	B,#0	; SET U	JP SUM POINTER
L00P:	RC LD	A,X		C CARRY TO START ADDITION POINTER TO ACC
	LAID	А,А		E VALUE FROM ROM TO ACC
	ADC	A,[B]	,	SUMLO TO ACC
	X CLR	A,[B+] A	; RESUI ; CLEAF	LT TO SUMLO B ACC
	ADC	A,[B]	; ADD S	SUMHI TO ACC
	X DBG7	A,[B-]		AT TO SUMHI
	DRSZ JP	X LOOP		AND TEST ROM PTR FOR ZERO BACK TO REPEAT LOOP
			; 11	F X PTR NOT ZERO
	SC	B,#2		F BORROW TO START SUBTRACTION JP TOT POINTER
LUP:	LD LD	□,#2 A,[X+]		RAHEND (SUM) TO ACC
	Х	A,[B]	; REVER	RSE OPERANDS
	SUBC X	A,[B] A,[B+]		DR SUBTRACTION LT TO TOT
	IFBNE	#4		TILL IN TOT FIELD
	JP	LUP		JMP BACK TO REPEAT LUP
	RET		; RETUR	RN FROM SUBROUTINE

2.0 MULTIPLICATION

The COP800 multiplications are all based on starting the multiplier in the low order end of the double length product space. The high end of the double length product space is initially cleared, and then the double length product is shifted right one bit. The bit shifted out from the low order end represents the low order bit of the multiplier. If this bit is a "1", the multiplicand is added to the high end of the double length product space. The entire shifting process and the conditional addition of the multiplicand to the upper end of the double length product is then repeated. The number of shift cycles is equal to the number of bit positions in the multiplier plus one extra shift cycle. This extra terminal shift cycle is necessary to correctly align the resultant product.

Note that an M byte multiplicand multiplied by an N byte multiplier will result in an M + N byte double length product. However, these multiplication subroutines will only use 2M + N + 1 bytes of RAM memory space, since the multiplier initially occupies the low order end of the double length product. The one extra byte is necessary for the shift counter CNTR.

The minimal code (28 byte) general multiplication subroutine is shown with two different examples, MY2448 and MY4824. Both examples multiply 24 bits by 48 bits. The MY2448 subroutine uses the 48-bit operand as the multiplier, and consequently uses minimal RAM as well as minimal program code. The MY4824 subroutine uses the 24-bit operand as the multiplier, and consequently executes considerably faster than the minimal RAM MY2448 subroutine.

- MPY88 - 8 by 8 Multiplication Subroutine - 19 Bytes - 180 Instruction Cycles
 - Minimum Code
- MLT88 - Fast 8 by 8 Multiplication Subroutine - 42 Bytes
 - 145 Instruction Cycles
- Very Fast 8 by 8 Multiply Subroutine VFM88 - 96 Bytes
 - 116 Instruction Cycles
- **MPY168** - Fast 16 by 8 Multiplication Subroutine - 36 Bytes
 - 230 Instruction Cycles Average
 - 254 Instruction Cycles Maximum

MPY816 (or MPY824, MPY832)

- 8 by 16 (or 24, 32) Multiply Subroutine
 - 22 Bytes
 - 589 (or 1065, 1669) Instruction Cycles Averade
 - 597 (or 1077, 1685) Instruction Cycles Maximum
 - Minimum Code, Minimum RAM
- Extendable Routine for MPY8XX by Changing Parameters, with Number of Bytes (22) Remaining a Constant - Fast 24 by 8 Multiplication Subroutine
- MPY248 - 47 Bytes
 - 289 Instruction Cycles Average
 - 333 Instruction Cycles Maximum
- MX1616 - Fast 16 by 16 Multiplication Subroutine - 39 Bytes
 - 498 Instruction Cycles Average
 - 546 Instruction Cycles Maximum
 - 16 by 16 Multiplicand Subroutine
- MP1616 - 29 Bytes
 - - 759 Instruction Cycles Average
 - 807 Instruction Cycles Maximum
 - Almost Minimum Code
- MY1616 (or MY1624, MY1632)
 - 28 Bytes
 - 16 by 16 (or 24, 32) Multiply Subroutine
 - 861 (or 1473, 2213) Inst. Cycles Average
 - 1029 (or 1725, 2549) Inst. Cycles Maxi
 - mum
 - Minimum Code, Minimum RAM
 - Extendable Routne for MY16XX by Changing Parameters, with Number of Bytes (28) Remaining a Constant

Minimal general multiplication subroutine for any number of bytes in multiplicand and multiplier

- 28 Bytes
- Minimum Code
- MY2448 Used as First Example, with Minimum RAM and 4713 Instruction Cycles Average 5457 Instruction Cycles Maximum
- MY4824 Used as Second Example, with Non Minimal RAM and 2751 Instruction Cycles Average 3483 Instruction Cycles Maximum

19 BYTES 180 INSTRUCTION CYCLES MULTIPLICAND IN [0] (ICAND) MULTIPLIER IN [1] (IER) FRODUCT IN [2,1] (PROD) MPY88: LD CNTR,#9 ; LD CNTR WITH LENGTH OF RC ; MULTIPLIER FIELD + 1 LD B,#2 CLR A ; CLEAR UPPER PRODUCT M88LUP: RRC A ; RIGHT SHIFT X A,[B-] ; UPPER PRODUCT LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B-] ; FRODUCT/MULTIPLIER CLR A ; CLR AC AND TEST LOW IFC ; ORDER MULTIPLICAND TO LD A,[B] ; UPPER PRODUCT LD A,[B] ; UPPER BIT LD A,[B] ; UPPER PRODUCT LD A,[B] ; UPPER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M88LUP ; CNTR FOR ZERO RET ; RETURN FROM SUBROUTINE		MINIMUM C	ODE			
MULTIPLICAND IN [0] (ICAND) MULTIPLIER IN [1] (IER) PRODUCT IN [2,1] (PROD) MPY88: LD CNTR,#9 ; LD CNTR WITH LENGTH OF RC ; MULTIPLIER FIELD + 1 LD B,#2						
MULTIPLIER IN [1] (IER) PRODUCT IN [2,1] (PROD) MPY88: LD CNTR,#9 ; LD CNTR WITH LENGTH OF RC ; MULTIPLIER FIELD + 1 LD B,#2 CLR A ; CLEAR UPPER PRODUCT M88LUP: RRC A ; RIGHT SHIFT X A,[B-] ; UPPER PRODUCT LD A,[B] ; RIGHT SHIFT LOWER RRC A ; RIGHT SHIFT LOWER ; LD A,[B] ; RIGHT SHIFT LOWER RC A ; CLR ACC AND TEST LOW ; IFC ; ORDER MULTIPLIER BIT ; ORDER MULTIPLER BIT LD A,[B] ; MULTIPLICAND TO ACC IF ; RC ; LOW ORDER BIT = 1 ; LD LD B,#2 ; ADD MULTIPLICAND TO ; ADC A,[B] ; UPPER PRODUCT ; DRSZ CNTR ; DECREMENT AND TEST ; JP M88LUP ; CNTR FOR ZERO ;						
PRODUCT IN $[2,1]$ (PROD)MPY88:LDCNTR,#9; LD CNTR WITH LENGTH OF RC; MULTIPLIER FIELD + 1LDB,#2;CLRA; CLEAR UPPER PRODUCTM88LUP:RRCA; RIGHT SHIFTXA,[B-]; UPPER PRODUCTLDA,[B];RRCA; RIGHT SHIFT LOWERXA,[B-]; PRODUCT/MULTIPLIERCLRA; CLR ACC AND TEST LOWIFC; ORDER MULTIPLER BITLDA,[B]; MULTIPLICAND TO ACC IFRC; LOW ORDER BIT = 1LDB,#2; ADD MULTIPLICAND TOADCA,[B]; UPPER PRODUCTDRSZCNTR; DECREMENT AND TESTJPM88LUP; CNTR FOR ZERO						
MPY88: LD CNTR,#9 ; LD CNTR WITH LENGTH OF RC ; MULTIPLIER FIELD + 1 LD B,#2 CLR A ; CLEAR UPPER PRODUCT M88LUP: RRC A ; CLEAR UPPER PRODUCT LD A,[B-] ; UPPER PRODUCT LD A,[B] ; RRC A RRC A ; RIGHT SHIFT LOWER X A,[B-] ; PRODUCT/MULTIPLIER CLR A ; CLR ACC AND TEST LOW IFC ; ORDER MULTIPLER BIT LD A,[B] ; MULTIPLICAND TO ACC IF RC ; LOW ORDER BIT = 1 LD B,#2 ; ADD MULTIPLICAND TO ADC A,[B] ; UPPER PRODUCT ADC A,[B] ; UPPER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M38LUP ; CNTF FOR ZERO						
RC ; MULTIPLIER FIELD + 1 LD B,#2 CLR A ; CLEAR UPPER PRODUCT M88LUP: RRC A ; RIGHT SHIFT X A,[B-] ; UPPER PRODUCT LD A,[B] RRC A RRC A ; RIGHT SHIFT LOWER X A,[B-] ; PRODUCT/MULTIPLIER CLR A ; CLR ACC AND TEST LOW IFC ; ORDER MULTIPLER BIT LD A,[B] ; MULTIPLICAND TO ACC IF RC ; LOW ORDER BIT = 1 LD B,#2 ; ADD MULTIPLICAND TO ADC A,[B] ; UPPER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M88LUP ; CNTR FOR ZERO					-	
LD B,#2 CLR A ; CLEAR UPPER PRODUCT M88LUP: RRC A ; RIGHT SHIFT X A,[B-] ; UPPER PRODUCT LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B-] ; PRODUCT/MULTIPLIER CLR A ; CLR ACC AND TEST LOW IFC ; ORDER MULTIPLER BIT LD A,[B] ; MULTIPLICAND TO ACC IF RC ; LOW ORDER BIT = 1 LD B,#2 ; ADD MULTIPLICAND TO ADC A,[B] ; UPPER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M88LUP ; CNTR FOR ZERO	MPY88:		CNTR,#9	,		
CLR A ; CLEAR UPPER PRODUCT M88LUP: RRC A ; RIGHT SHIFT X A,[B-] <td; product<="" td="" upper=""> LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B-] ; PRODUCT/MULTIPLIER CLR A ; CLR ACC AND TEST LOW IFC ; ORDER MULTIPLER BIT LD A,[B] ; MULTIPLICAND TO ACC IF RC ; LOW ORDER BIT = 1 LD B,#2 ; ADD MULTIPLICAND TO ADC A,[B] ; UPPER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M88LUP ; CNTR FOR ZERO</td;>				;	MULTIPLIER FIELD + 1	
M88LUP: RRC A ; RIGHT SHIFT X A,[B-] ; UPPER PRODUCT LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B-] ; PRODUCT/MULTIPLIER CLR A ; CLR ACC AND TEST LOW IFC ; ORDER MULTIPLER BIT LD A,[B] ; MULTIPLICAND TO ACC IF RC ; LOW ORDER BIT = 1 LD B,#2 ; ADD MULTIPLICAND TO ADC A,[B] ; UPPER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M88LUP ; CNTR FOR ZERO		-				
XA,[B-];UPPER PRODUCTLDA,[B]RRCA;RIGHT SHIFT LOWERXA,[B-];PRODUCT/MULTIPLIERCLRA;CLR ACC AND TEST LOWIFC;ORDER MULTIPLER BITLDA,[B];MULTIPLICAND TO ACC IFRC;LOW ORDER BIT = 1LDB,#2;ADD MULTIPLICAND TOADCA,[B];UPPER PRODUCTDRSZCNTR;DECREMENT AND TESTJPM88LUP;CNTR FOR ZERO						
LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B-] ; PRODUCT/MULTIPLIER CLR A ; CLR ACC AND TEST LOW IFC ; ORDER MULTIPLER BIT LD A,[B] ; MULTIPLICAND TO ACC IF RC ; LOW ORDER BIT = 1 LD B,#2 ; ADD MULTIPLICAND TO ADC A,[B] ; UPPER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M88LUP ; CNTR FOR ZERO	M88LUP:	RRC				
RRCA;RIGHT SHIFT LOWERXA,[B-];PRODUCT/MULTIPLIERCLRA;CLR ACC AND TEST LOWIFC;ORDER MULTIPLER BITLDA,[B];MULTIPLICAND TO ACC IFRC;LOW ORDER BIT = 1LDB,#2;ADD MULTIPLICAND TOADCA,[B];UPPER PRODUCTDRSZCNTR;DECREMENT AND TESTJPM88LUP;CNTR FOR ZERO		Х	A,[B—]	;	UPPER PRODUCT	
XA,[B-];PRODUCT/MULTIPLIERCLRA;CLR ACC AND TEST LOWIFC;ORDER MULTIPLER BITLDA,[B];MULTIPLICAND TO ACC IFRC;LOW ORDER BIT = 1LDB,#2;ADD MULTIPLICAND TOADCA,[B];UPPER PRODUCTDRSZCNTR;DECREMENT AND TESTJPM88LUP;CNTR FOR ZERO		LD				
CLRA; CLR ACC AND TEST LOWIFC; ORDER MULTIPLER BITLDA,[B]RC; LOW ORDER BIT = 1LDB,#2ADCA,[B]UPPER FRODUCTDRSZCNTRJPM88LUPCLRCLR ACC AND TEST LOW		RRC				
IFC;ORDER MULTIPLER BITLDA,[B];MULTIPLICAND TO ACC IFRC;LOW ORDER BIT = 1LDB,#2;ADD MULTIPLICAND TOADCA,[B];UPPER PRODUCTDRSZCNTR;DECREMENT AND TESTJPM88LUP;CNTR FOR ZERO		Х	A,[B—]	;	PRODUCT/MULTIPLIER	
LDA,[B];MULTIPLICAND TO ACC IFRC;LOW ORDER BIT = 1LDB,#2;ADD MULTIPLICAND TOADCA,[B];UPPER PRODUCTDRSZCNTR;DECREMENT AND TESTJPM88LUP;CNTR FOR ZERO		CLR	A	;	CLR ACC AND TEST LOW	
RC;LOW ORDER BIT = 1LDB,#2;ADD MULTIPLICAND TOADCA,[B];UPPER PRODUCTDRSZCNTR;DECREMENT AND TESTJPM88LUP;CNTR FOR ZERO		IFC		;		
LD B,#2 ; ADD MULTIPLICAND TO ADC A,[B] ; UPPER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M88LUP ; CNTR FOR ZERO		LD	A,[B]			
ADC A,[B] ; UPPER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M88LUP ; CNTR FOR ZERO		RC		;	LOW ORDER BIT = 1	
ADC A,[B] ; UPPER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M88LUP ; CNTR FOR ZERO		-	B,#2	;	ADD MULTIPLICAND TO	
JP M88LUP ; CNTR FOR ZERO			A,[B]	;	UPPER PRODUCT	
		DRSZ				
RET ; RETURN FROM SUBROUTINE		JP	M88LUP	;	CNTR FOR ZERO	
		RET		;	RETURN FROM SUBROUTINE	

MLT88—FAST 8 BY 8 MULTIPLICATION SUBROUTINE

	42 BYTES 145 INSTRUC	TION CYCLES		
	MULTIPLICAN	D IN FOI	(ICAND)	
	MULTIPLIER		(IER)	
	PRODUCT IN		(PROD)	
MLT88:	LD	CNTR,#3		
MT100:		CNIR,#5		
	RC	B //O	; 1/3 OF LENGTH OF	
	LD	B,#2	; (MULTIPLIER FIELD + 1)	
	CLR	A	; CLEAR UPPER PRODUCT	
;	DDd			
ML88LP:	RRC	A	; RIGHT SHIFT ***	
	X	A,[B-]	; UPPER PRODUCT	
	LD	A,[B]		
	RRC	A	; RIGHT SHIFT LOWER	
	Х	A,[B-]	; PRODUCT/MULTIPLIER	
	CLR	A	; CLR ACC AND TEST LOW	
	IFC		; ORDER MULTIPLIER BIT	
	LD	A,[B]	; MULTIPLICAND TO ACC IF	
	RC		; LOW ORDER BIT = 1	
	LD	B,#2	; ADD MULTIPLICAND TO	
	ADC	A,[B]	; UPPER PRODUCT ***	
;				
	RRC	A	; REPEAT THE ABOVE	
	Х	A,[B-]	; 11 BYTE	
	LD	A,[B]	; 13 INSTRUCTION	
	RRC	A	; CYCLE PROGRAM	
	Х	A,[B-]	; SECTION (WITH	
	CLR	A	; THE *** DELIMITERS)	
	IFC		; TWICE MORE FOR A	
	LD	A,[B]	TOTAL OF THREE TIMES	
	RC	, L- J	,	
	LD	B,#2		
	ADC	A,[B]	: END OF SECOND REPEAT	
;		,[2]	, 2.2 Of DECOMP HEIERI	
,	RRC	A	: START OF THIRD REPEAT	
	X	A,[B-]	, SIMILOF IMIND MELERI	
	LD	A,[B]		
	RRC	А,[D] А		
	X			
		A,[B-]		
	CLR	A		
	IFC	4 501		
	LD	A,[B]		
	RC	D #0		
	LD	B,#2		
	ADC	A,[B]	; END OF THIRD REPEAT	
;	2244	a		
	DRSZ	CNTR	; DECREMENT AND TEST	
	JMP	ML88LP	; CNTR FOR ZERO	
	RET		; RETURN FROM SUBROUTINE	

	96 BYTES 116 INST	RUCTION CYCLES		
	MULTIPLI	CAND IN [0] ER IN [1] IN [2,1]	(ICAND) (IER) (PROD)	
/FM88:	RC	1. [2,1]		
1.1100.	LD	B,#2		
	LD	[B-],#0	; CLEAR UPPER PRODUCT	
	LD	Δ [B]		
	RRC	A A	; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
	X	A,[B-]	· PRODUCT /MILTIPLIER	
	CLR	A,[D-] A	; CLR ACC AND TEST LOW	
	IFC	А	; ORDER MULTIPLIER BIT	
	LD	A,[B]	-	
	RC	м,[D]	· LOW ORDER RIT - 1	
	LD	B,#2	· ADD MULTIPLICAND TO	
	ADC	A,[B]	; MULTIPLICAND TO ACC IF ; LOW ORDER BIT = 1 ; ADD MULTIPLICAND TO ; UPPER PRODUCT	
	ADO	м,[D]	, CITER INDUCT	
;	RRC	А	; RIGHT SHIFT ***	
	X	A,[B-]	; UPPER PRODUCT	
	LD	A,[B]	, orda radio or	
	RRC	A A	• RICHT SHIFT LOWER	
	X	A,[B-]	; PRODUCT/MULTIPLIER	
	CLR	A A	; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER ; CLR ACC AND TEST LOW	
	IFC	11	; ORDER MULTIPLIER BIT	
	LD	A.[B]	; MULTIPLICAND TO ACC IF	
	RC	n,[b]	; LOW ORDER BIT = 1	
	LD	B,#2	; ADD MULTIPLICAND TO	
	ADC	A,[B]	; UPPER PRODUCT ***	
DEL			CTION CYCLE SECTION WITH THE *** OCESSING FOR ONE MULTIPLIER BIT.	
DEL			OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE	
DEL			OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION	
DEL			OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES,	
DEL			OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION	
DEL	JIMITERS RE	PRESENTS THE PRO	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES	
DEL	JIMITERS RE	PRESENTS THE PRO	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT	
DEL	JIMITERS RE	A A,[B-]	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT	
DEL	LIMITERS RE	A A,[B-] A,[B]	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT	
DEL	JIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT	
DEL	LIMITERS RE	A A,[B-] A,[B]	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER	
; DEL ; ; ; ;	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
; DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
; DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	
; DEL	LIMITERS RE	A A,[B-] A,[B] A	OCESSING FOR ONE MULTIPLIER BIT. ; REPEAT THE ; ABOVE SECTION ; SIX MORE TIMES, ; FOR A TOTAL ; OF SEVEN TIMES ; RIGHT SHIFT ; UPPER PRODUCT ; RIGHT SHIFT LOWER ; PRODUCT/MULTIPLIER	

NULTIFLICAND IN [1,0] (ICAND) MULTIFLIER IN [2] (IER) PRODUCT IN [4,3,2] (PROD) MEYIAGS LD CNTR,#9 : LD CNTR WITH LENGTH OF RC : MULTIFLIER FIELD + 1 LD B,#4 LD [B-1,#0] : CLEAR LD [B-1,#0] : UFPER FRODUCT JF MF168S RIGHT SHIFT IDDLE X A,[B-1] : BYIE OF FRODUCT LD A,[B] : RIGHT SHIFT LOWER X A,[B] : RIGHT SHIFT LOWER X A,[B] : PRODUCT/MULTIFLIER IFNC : TEST LOWER BIT OF WILTIFLICAND TO ACC LD B,#0 : LOWER BYTE OF CLD LD B,#3 : ADD LOWER BYTE OF FROD LD B,#4 : MULTIFLICAND TO ACC LD LD B,#1 : MULTIFLICAND TO ACC LD A,[B] : MDC A,[B] : MULTIFLICAND TO ACC	MULTIPLIER IN [2] (IER) FRODUCT IN [4,3,2] (PROD) APY168: LD CNTR,#9 : LD CNTR WITH LENGTH OF RC ; MULTIPLIER FIELD + 1 LD B,#4 LD [B-],#0 : CLEAR LD [B-],#0 ; UPPER PRODUCT JP MP168S AL68LF: RRC A ; RIGHT SHIFT UPPER X A,[B-] ; BYTE OF PRODUCT LD A,[B] RRC A ; RIGHT SHIFT MIDDLE X A,[B-] ; BYTE OF PRODUCT IFNC ; TEST LOWER BIT JF MP168T ; OF MULTIPLIER RC A ; RIGHT SHIFT LOWER X A,[B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JF MP168T ; OF MULTIPLIER RC A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD LD B,#4 ; ADD UPPER BYTE OF FROD LD B,#4 ; ADD UPPER BYTE OF FROD LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD LD B,#4 ; ADD UPPER BYTE OF FROD LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD LD B,#4 ; ADD UPPER BYTE OF FROD LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JF MIG8LP ; BACK TO LOOC; CNTR (ANNOT EQUAL ZERO MP168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; MULTER EQUAL TO ZERO			RUCTION CYCLES AV RUCTION CYCLES MA	
IPY168:LDCNTR,#9:LD CNTR WITH LENGTH OF RCRC;MULTIPLIER FIELD + 1LDB,#4LD[B-],#0;CLEARLD[B-],#0;UPPER PRODUCTJFMP168SN168LP:RRCAXA,[B-];BYTE OF PRODUCTLDA,[B]RRCAXA,[B-]EP168S:LDA,[B]RRCAXA,[B]RRCAXA,[B]RRCARRCARRCARRCARRCARRCARRCA,[B]RRCA,[B]FIFNC;CLEAR CARRYLDB,#0LDA,[B]MULTIPLICAND TO ACCLDA,[B]MULTIPLICAND TOLDA,[B]MULTIPLICAND TOXA,[B]MULTIPLICAND TOXA,[B]MULTIPLICAND TO ACCLDB,#1UPPER BYTE OF FRODLDA,[B]TO UPPER BYTE	PY168:LDCNTR,#9:LD CNTR WITH LENGTH OFRC;MULTIPLIER FIELD + 1LDB,#4LD[B-],#0:LD[B-],#0;UPFER PRODUCTJPMP168SNE68E:N168LP:RRCXA,[B-]RRCAXA,[B-]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]IFNC:CDB,#0LDB,#1UPER BYTE OFLDA,[B]MULTIPLICAND TO ACCLDB,#1UPPER BYTE OFLDA,[B]MICSLP:CANNOT EQUAL DO PRODDRSZCMTRCMTR:DRSZCMTRCMTR:DRSZCMTRCMTR:<		MULTIPLI	ER IN [2]	(IER)
RC;MULTIPLIER FIELD + 1LD $B,\#4$.LD $[B-],\#0$;CLEAR.LD $[B-],\#0$;UPPER PRODUCTJPMP168SIL68LF:RRCAXA, $[B-]$ RRCAXA, $[B-]$ RRCAXA, $[B]$ RRCAAB,#0IP168S:LDLDB,#0IDB,#3GCLEAR CARRYLDB,#3ADDLOWER BYTE OFLDA, $[B]$ MULTIPLICAND TO ACCLDB,#3ADDLOWER BYTE OF FRODLDB,#1UPPER BYTE OF FRODLDB,#4ADDADCA, $[B]$ MICHTPICAND TO ACCLDB,#4ADDADCA, $[B]$ TOUPPER BYTE OF FRODDRSZCNTRCONNOT EQUAL ZEROP168T:LDA, $[B]$ SZ2CNTRCNTREQUAL TO ZERO	RC;MULTIPLIER FIELD + 1LDB,#4LD[B-],#0;CLEARLD[B-],#0;UPPER PRODUCTJPMP168S168LP:RRCAXA,[B-]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]RRCAXA,[B]P168S:LDLDA,[B]RRCAXA,[B]RRCAXA,[B]P168S:LDLDA,[B]RRCAXA,[B]RRCARRCCLEAR CARRYLDB,#0LDS,#0LDB,#3ADDLOWER BYTE OFLDA,[B]MULTIFLICAND TO ACCLDB,#1LDB,#1LDB,#4ADCA,[B]MLTIFLICAND TO ACCLDB,#4ADDUPPER BYTE OF PRODLDB,#4ADCA,[B]CANNOT EQUAL ZEROP168T:LDLDB,#4HIGH ORDER PRODUCTLDA,[B]BACK TO LOOP; CNTRCANNOT EQUAL ZEROP168T:LDLDA,[B]F168T:LDLD <td< th=""><th>PV169.</th><th></th><th></th><th></th></td<>	PV169.			
LD B,#4 LD [B-],#0 ; CLEAR LD [B-],#0 ; UPPER PRODUCT JP MP168S (168LP: RRC A ; RIGHT SHIFT UPPER X A,[B-] ; BYTE OF PRODUCT LD A,[B] RRC A ; RIGHT SHIFT MIDDLE X A,[B-] ; BYTE OF PRODUCT P168S: LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B-] ; BYTE OF PRODUCT IP168S: LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF FROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD JF M168LP ; CNTR EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER FRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	LD B,#4 LD [B-],#0 ; CLEAR LD [B-],#0 ; UPPER PRODUCT JP MP168S (168LP: RRC A ; RIGHT SHIFT UPPER X A,[B-] ; BYTE OF PRODUCT LD A,[B] RRC A ; RIGHT SHIFT MIDDLE X A,[B-] ; BYTE OF PRODUCT P168S: LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B-] ; BYTE OF PRODUCT IFINC : TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC : CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF FROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD JF M168LP ; CANNOT EQUAL ZERO (ANNOT EQUAL ZERO DRSZ CNTR ; DECREMENT AND TEST IF JF M168LP ; CNTR EQUAL TO ZERO	11100:		CMIR,#9	
LD $[B-], \#0$; CLEAR LD $[B-], \#0$; UPPER FRODUCT JP MP168S 168LP: RRC A : RIGHT SHIFT UPPER X A, [B-] ; BYTE OF PRODUCT LD A, [B] RRC A ; RIGHT SHIFT MIDDLE X A, [B-] ; BYTE OF PRODUCT P168S: LD A, [B] RRC A ; RIGHT SHIFT LOWER X A, [B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B, #0 ; LOWER BYTE OF LD A, [B] ; MULTIPLIER RC ; CLEAR CARRY LD B, #3 ; ADD LOWER BYTE OF LD A, [B] ; MULTIPLICAND TO ACC LD B, #3 ; ADD LOWER BYTE OF ADC A, [B] ; MULTIPLICAND TO ACC LD B, #1 ; UPPER BYTE OF LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF FROD LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF FROD LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF FROD LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF FROD LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF FROD LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF FROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; CNTR EQUAL ZERO P168T: LD B, #4 ; HIGH ORDER PRODUCT LD A, [B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	LD $[B-], \#0$; CLEAR LD $[B-], \#0$; UPPER PRODUCT JP MP168S 168LP: RRC A ; RIGHT SHIFT UPPER X A, [B-] ; BYTE OF PRODUCT LD A, [B] RRC A ; RIGHT SHIFT MIDDLE X A, [B-] ; BYTE OF PRODUCT P168S: LD A, [B] RRC A ; RIGHT SHIFT LOWER X A, [B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B, #0 ; LOWER BYTE OF LD A, [B] ; MULTIPLICAND TO ACC LD B, #3 ; ADD LOWER BYTE OF ADC A, [B] ; MULTIPLICAND TO ACC LD B, #3 ; ADD LOWER BYTE OF ADC A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; UPPER BYTE OF FROD LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; OF MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF FROD LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF FROD LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF FROD LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF FROD LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF FROD LD A, [B] ; MULTIPLICAND TO ACC LD B, #4 ; ADD UPPER BYTE OF FROD JP MI68LP ; CNTR EQUAL ZERO P168T: LD B, #4 ; HIGH ORDER PRODUCT LD A, [B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP MI68LP ; CNTR EQUAL TO ZERO			B,#4	,
JP MP168S 168LP: RRC A ; RIGHT SHIFT UPPER X A,[B-] ; BYTE OF PRODUCT LD A,[B] RRC A ; RIGHT SHIFT MIDDLE X A,[B-] ; BYTE OF PRODUCT P168S: LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MIDDLE BYTE OF FROD LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF P168T ; DE B,#4 ; ADD UPPER BYTE OF FROD LD B,#4 ; ADD UPPER BYTE OF FROD P168T; LD B,#4 ; HIGH ORDER FRODUCT LD B,#4 ; HIGH ORDER FRODUCT JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T; LD B,#4 ; HIGH ORDER FRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	JP MP168S 168LP: RRC A ; RIGHT SHIFT UPPER X A,[B-] ; BYTE OF PRODUCT LD A,[B] RRC A ; RIGHT SHIFT MIDDLE X A,[B-] ; BYTE OF PRODUCT P168S: LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF PROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF PROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF PROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF PROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF PROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF PROD P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; DYPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; DYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO		LD		; CLEAR
168LP: RRC A ; RIGHT SHIFT UPPER X A,[B-] ; BYTE OF PRODUCT LD A,[B]	168LP:RRCA;RIGHT SHIFT UPPERXA,[B-];BYTE OF FRODUCTLDA,[B];RIGHT SHIFT MIDDLEXA,[B-];BYTE OF FRODUCTP168S:LDA,[B];RRCA;RIGHT SHIFT LOWERXA,[B];PRODUCT/MULTIPLIERIFNC;TEST LOWER BITJPMP168T;Of MULTIPLIERRC;CLEAR CARRYLDB,#0;LOWER BYTE OFLDA,[B];MULTIPLICAND TO ACCLDB,#3;ADD LOWER BYTE OFLDB,#3;ADD LOWER BYTE OF FRODLDB,#1;UPPER BYTE OF FRODLDB,#1;UPPER BYTE OF FRODLDB,#4;ADD UPPER BYTE OF FRODDRSZCNTR;DECREMENT CNTR AND JUMPJPM168LP;BACK TO LOOP ; CNTRLDA,[B];HIGH ORDER PRODUCTJPM168LP;BACK TO LOOP ; CNTRJPM168LP;BACK TO LOOP ; CNTRJPM168LP;BATE TO ACCDRSZ		LD	[B-],#O	; UPPER PRODUCT
X A,[B-]; BYTE OF PRODUCT LD A,[B] RRC A ; RIGHT SHIFT MIDDLE X A,[B-]; BYTE OF PRODUCT P168S: LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B]; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF FROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF FROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF FROD LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	X A, [B-]; BYTE OF PRODUCT LD A, [B] RRC A ; RIGHT SHIFT MIDDLE X A, [B-]; BYTE OF PRODUCT P168S: LD A, [B] RRC A ; RIGHT SHIFT LOWER X A, [B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A, [B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A, [B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF PROD LD A, [B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF PROD LD A, [B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF PROD LD A, [B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF PROD P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A, [B] ; TO UPPER BYTE OF PROD P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A, [B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A, [B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO				
LD A,[B] RRC A ; RIGHT SHIFT MIDDLE X A,[B-] ; BYTE OF FRODUCT P168S: LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF FROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER FRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	LD A,[B] RRC A ; RIGHT SHIFT MIDDLE X A,[B-] ; BYTE OF PRODUCT P168S: LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO ADC A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	168LP:			
RRCA;RIGHT SHIFT MIDDLEXA,[B-];BYTE OF PRODUCTP168S:LDA,[B]RRCA;RIGHT SHIFT LOWERXA,[B];PRODUCT/MULTIPLIERIFNC;TEST LOWER BITJPMP168T;OF MULTIPLIERRC;CLEAR CARRYLDB,#0;LOWER BYTE OFLDA,[B];MULTIPLICAND TO ACCLDB,#3;ADD LOWER BYTE OFADCA,[B];MIDDLE BYTE OF PRODLDB,#1;UPPER BYTE OFLDA,[B];MULTIPLICAND TO ACCLDB,#1;UPPER BYTE OF FRODLDB,#1;UPPER BYTE OF FRODLDB,#4;ADD UPPER BYTE OF FRODDRSZCNTR;DECREMENT CNTR AND JUMPJPM168LP;BACK TO LOOP; CNTRLDA,[B];BYTE TO ACCDRSZCNTR;DETMENT AND TEST IFJPM168LP;CNTR EQUAL TO ZERO	RRCA;RIGHT SHIFT MIDDLEXA,[B-];BYTE OF PRODUCTP168S:LDA,[B]RRCA;RIGHT SHIFT LOWERXA,[B];PRODUCT/MULTIPLIERIFNC;TEST LOWER BITJPMP168T;OF MULTIPLIERRC;CLEAR CARRYLDB,#0;LOWER BYTE OFLDA,[B];MULTIPLICAND TO ACCLDB,#3;ADD LOWER BYTE OFADCA,[B];MULTIPLICAND TOXA,[B];MULTIPLICAND TO ACCLDB,#1;UPPER BYTE OF PRODLDB,#1;UPPER BYTE OFLDA,[B];MULTIPLICAND TO ACCLDB,#4;ADD UPPER BYTE OF FRODDRSZCNTR;DECREMENT CNTR AND JUMPJPM168LP;BACK TO LOOP; CNTRLDA,[B];BYTE TO ACCDRSZCNTR;DECREMENT AND TEST IFJPM168LP;CNTR EQUAL TO ZERO				; BYTE OF PRODUCT
X A,[B-] ; BYTE OF PRODUCT P168S: LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER FRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	X A,[B-] ; BYTE OF PRODUCT P168S: LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF FROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO				
P168S: LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO ; P168T: LD B,#4 ; HIGH ORDER FRODUCT LD A,[E] ; BYTE TO ACC DRSZ CNTR ; DECR	P168S: LD A,[B] RRC A ; RIGHT SHIFT LOWER X A,[B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#1 ; UPPER BYTE OF FROD LD A,[B] ; MULTIPLICAND ADC A,[B] ; TO UPPER BYTE OF FROD DRSZ CMTR ; DECREMENT CNTR AND JUMP JP MI68LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO <				
RRCA;RIGHT SHIFT LOWERXA,[B];PRODUCT/MULTIPLIERIFNC;TEST LOWER BITJPMP168T;OF MULTIPLIERRC;CLEAR CARRYLDB,#0;LOWER BYTE OFLDA,[B];MULTIPLICAND TO ACCLDB,#3;ADD LOWER BYTE OFADCA,[B];MULTIPLICAND TOXA,[B];MULTIPLICAND TOLDB,#1;UPPER BYTE OF FRODLDB,#1;MULTIPLICAND TO ACCLDB,#4;ADD UPPER BYTE OF ICANDADCA,[B];TO UPPER BYTE OF PRODDRSZCNTR;DECREMENT CNTR AND JUMPJPM168LP;BACK TO LOOP ; CNTRCANNOT EQUAL ZERO;CANNOT EQUAL ZEROP168T:LDB,#4;HIGH ORDER PRODUCTLDA,[B]JPM168LP;CNTR;DECREMENT AND TEST IFJPM168LP;CNTR EQUAL TO ZERO	RRCA; RIGHT SHIFT LOWERXA,[B]; PRODUCT/MULTIPLIERIFNC; TEST LOWER BITJPMP168T; OF MULTIPLIERRC; CLEAR CARRYLDB,#0; LOWER BYTE OFLDA,[B]; MULTIPLICAND TO ACCLDB,#3; ADD LOWER BYTE OFADCA,[B]; MULTIPLICAND TO ACCLDB,#3; ADD LOWER BYTE OFLDB,#1; UPPER BYTE OFLDA,[B]; MULTIPLICAND TO ACCLDB,#4; ADD UPPER BYTE OF FRODLDB,#4; DECREMENT CNTR AND JUMPJPM168LP; BACK TO LOOP; CNTRJPM168LP; BYTE TO ACCDRSZCNTR; DECREMENT AND TEST IFJPM168LP; TO UPPER BYTE TO ACCDRSZCNTR; DECREMENT AND TEST IFJPM168LP; CNTR EQUAL TO ZERO	P1685.			; BILE OF FRODUCI
X A,[B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO X A,[B] ; MIDDLE BYTE OF PROD LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER FRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	X A,[B] ; PRODUCT/MULTIPLIER IFNC ; TEST LOWER BIT JP MP168T ; OF MULTIPLIER RC ; CLEAR CARRY LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF FROD LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF FROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	1005.			: RIGHT SHIFT LOWER
IFNC; TEST LOWER BITJPMP168T; OF MULTIPLIERRC; CLEAR CARRYLDB,#0; LOWER BYTE OFLDA,[B]; MULTIPLICAND TO ACCLDB,#3; ADD LOWER BYTE OFADCA,[B]; MULTIPLICAND TOXA,[B]; MULTIPLICAND TOLDB,#1; UPPER BYTE OF PRODLDB,#1; UPPER BYTE OF ICANDADCA,[B]; MULTIPLICAND TO ACCLDB,#1; UPPER BYTE OF ICANDADCA,[B]; TO UPPER BYTE OF ICANDADCA,[B]; TO UPPER BYTE OF PRODDRSZCNTR; DECREMENT CNTR AND JUMPJPM168LP; BACK TO LOOP; CNTRLDB,#4; HIGH ORDER PRODUCTLDA,[B]; BYTE TO ACCDRSZCNTR; DECREMENT AND TEST IFJPM168LP; CNTR EQUAL TO ZERO	IFNC; TEST LOWER BITJPMP168T; OF MULTIPLIERRC; CLEAR CARRYLDB,#0; LOWER BYTE OFLDA,[B]; MULTIPLICAND TO ACCLDB,#3; ADD LOWER BYTE OFADCA,[B]; MULTIPLICAND TOXA,[B]; MULTIPLICAND TOLDB,#1; UPPER BYTE OFLDA,[B]; MULTIPLICAND TO ACCLDB,#4; UPPER BYTE OF ICANDLDB,#4; ADD UPPER BYTE OF ICANDADCA,[B]; TO UPPER BYTE OF ICANDADCA,[B]; DECREMENT CNTR AND JUMPJPM168LP; BACK TO LOOP; CNTRLDB,#4; HIGH ORDER PRODUCTLDA,[B]; BYTE TO ACCDRSZCNTR; DECREMENT AND TEST IFJPM168LP; CNTR EQUAL TO ZERO				
RC; CLEAR CARRYLDB,#0; LOWER BYTE OFLDA,[B]; MULTIPLICAND TO ACCLDB,#3; ADD LOWER BYTE OFADCA,[B]; MULTIPLICAND TOXA,[B]; MULTIPLICAND TOLDB,#1; UPPER BYTE OFLDA,[B]; MULTIPLICAND TO ACCLDB,#1; UPPER BYTE OFLDA,[B]; MULTIPLICAND TO ACCLDB,#4; ADD UPPER BYTE OF FRODDRSZCNTR; DECREMENT CNTR AND JUMPJPM168LP; BACK TO LOOP; CNTRLDA,[B]; BYTE TO ACCDRSZCNTR; DECREMENT AND TEST IFJPM168LP; DYTE TO ACCDRSZCNTR; DECREMENT AND TEST IFJPM168LP; CNTR EQUAL TO ZERO	RC; CLEAR CARRYLDB,#0; LOWER BYTE OFLDA,[B]; MULTIPLICAND TO ACCLDB,#3; ADD LOWER BYTE OFADCA,[B]; MULTIPLICAND TOXA,[B]; MULTIPLICAND TOLDB,#1; UPPER BYTE OFLDA,[B]; MULTIPLICAND TO ACCLDB,#1; UPPER BYTE OFLDA,[B]; MULTIPLICAND TO ACCLDB,#4; ADD UPPER BYTE OF ICANDADCA,[B]; TO UPPER BYTE OF PRODDRSZCMTR; DECREMENT CNTR AND JUMPJPM168LP; BACK TO LOOP; CNTRCANNOT EQUAL ZERO;P168T:LDB,#4HIGH ORDER PRODUCT;LDA,[B]; BYTE TO ACCDRSZCNTR; DECREMENT AND TEST IFJPM168LP; CNTR EQUAL TO ZERO				
LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	LD B,#0 ; LOWER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MIDDLE BYTE OF PROD LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO		JP	MP168T	; OF MULTIPLIER
LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MULTIPLICAND TO LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	LD A,[B] ; MULTIPLICAND TO ACC LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MIDTIPLICAND TO X A,[B] ; MIDDLE BYTE OF PROD LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP MI68LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP MI68LP ; CNTR EQUAL TO ZERO				
LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MIDDLE BYTE OF PROD LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	LD B,#3 ; ADD LOWER BYTE OF ADC A,[B] ; MULTIPLICAND TO X A,[B] ; MIDDLE BYTE OF PROD LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO				
ADCA,[B];MULTIPLICAND TOXA,[B];MIDDLE BYTE OF PRODLDB,#1;UPPER BYTE OFLDA,[B];MULTIPLICAND TO ACCLDB,#4;ADD UPPER BYTE OF ICANDADCA,[B];TO UPPER BYTE OF FRODDRSZCNTR;DECREMENT CNTR AND JUMPJPM168LP;BACK TO LOOP; CNTRCANNOT EQUAL ZERO;CANNOT EQUAL ZEROP168T:LDB,#4;HIGH ORDER PRODUCTLDA,[B];BYTE TO ACCDRSZCNTR;DECREMENT AND TEST IFJPM168LP;CNTR EQUAL TO ZERO	ADCA,[B];MULTIPLICAND TOXA,[B];MIDDLE BYTE OF PRODLDB,#1;UPPER BYTE OFLDA,[B];MULTIPLICAND TO ACCLDB,#4;ADD UPPER BYTE OF ICANDADCA,[B];TO UPPER BYTE OF PRODDRSZCNTR;DECREMENT CNTR AND JUMPJPM168LP;BACK TO LOOP; CNTR:CANNOT EQUAL ZEROP168T:LDB,#4;LDA,[B];BYTE TO ACCDRSZCNTR;DECREMENT AND TEST IFJPM168LP;CNTR EQUAL TO ZERO				
X A,[B] ; MIDDLE BYTE OF PROD LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	X A, [B] ; MIDDLE BYTE OF PROD LD B,#1 ; UPPER BYTE OF LD A, [B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A, [B] ; TO UPPER BYTE OF PROD DRSZ CMTR ; DECREMENT CNTR AND JUMP JP MI68LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A, [B] ; BYTE TO ACC DRSZ CMTR ; DECREMENT AND TEST IF JP MI68LP ; CNTR EQUAL TO ZERO				
LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF FROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER FRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	LD B,#1 ; UPPER BYTE OF LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO				,
LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	LD A,[B] ; MULTIPLICAND TO ACC LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CMTR ; DECREMENT CMTR AND JUMP JP M168LP ; BACK TO LOOP; CMTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CMTR ; DECREMENT AND TEST IF JP M168LP ; CMTR EQUAL TO ZERO				
LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	LD B,#4 ; ADD UPPER BYTE OF ICAND ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO				
ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	ADC A,[B] ; TO UPPER BYTE OF PROD DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO				
DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	DRSZ CNTR ; DECREMENT CNTR AND JUMP JP M168LP ; BACK TO LOOP; CNTR ; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO				
; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	; CANNOT EQUAL ZERO P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO		DRSZ		
P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	P168T: LD B,#4 ; HIGH ORDER PRODUCT LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO		JP	M168LP	
LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	LD A,[B] ; BYTE TO ACC DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO				
DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	DRSZ CNTR ; DECREMENT AND TEST IF JP M168LP ; CNTR EQUAL TO ZERO	P168T:			
JP M168LP ; CNTR EQUAL TO ZERO	JP M168LP ; CNTR EQUAL TO ZERO				
				MICOL	

PARAMETE	077, 1685) INSTR. CY E ROUTINE FOR MPY8XX RS, WITH NUMBER OF E G A CONSTANT.	K BY CHAN	GING
	R IN [2,1] FOR 16 BI OR [3,2,1] for 2	IT (IER) 24 BIT	ND)
PRODUCT I	N [3,2,1] FOR 16 BIT OR [4,3,2,1] FOF	r (PRO R 24 BIT	
LD	CNTR,#17	;;;	
RC		,	
LD	B,#3	;	#3 FOR MPY816 (#4 FOR MPY824) (#5 FOR MPY832)
LD	[B-],#O		CLEAR UPPER PRODUCT
LD	A,[B]	-	FIVE INSTRUCTION
			RIGHT SHIFT PRODUCT/MULTIPLIER
JP	#0 M8XXLP		LOOP JUMP BACK
CLR	A		CLR ACC AND TEST LOW
IFNC			
	M8XXT	;	JP IF LOW ORDER BIT = 0
	B #0		
		:	MULTIPLICAND TO ACC
LD	B,#3		#3 FOR MPY816
		;	(#4 FOR MPY824)
150			
ADC	А,[В]	;	ADD MULTIPLICAND TO UPPER BYTE OF PRODUCT
DRSZ	CNTR	;	DECREMENT AND TEST
JP	M8XXL	;	CNTR FOR ZERO
RET		;	RETURN FROM SUBROUTINE
	REMAINING MULTIPLICA MULTIPLIED PRODUCT IN LD LD LD LD LD LD LD LD LD LD LD LD LD	REMAINING A CONSTANT.MULTIPLICAND IN [0] MULTIPLIER IN [2,1] FOR 16 BI OR [3,2,1] FOR 16 BI OR [4,3,2,1] FOR 16 BI OR [5,4,3,2,1] FOR OR [5,4,3,2,1] FOR OR [5,4,3,2,1] FOR DELDCNTR,#17LDCNTR,#17LD $[B-],#0$ LDLD $[B-],#0$ LDLD $[B-],#0$ LDLD $[B-],#0$ LDLD $[B-],#0$ LDLD $[B-],#0$ LDLD $[B-],#0$ LDLD $[B-],#0$ LDLD $[B,#3]$ CLRA IFBNEJPM8XXLP CLRCLRA IFNCJPM8XXT RC LDLD $B,#0$ LDLD $B,#3$ ADC $A, [B]$ DRSZDRSZCNTR M8XXL	REMAINING A CONSTANT. MULTIPLICAND IN [0] (ICA MULTIPLIER IN [2,1] FOR 16 BIT (IER) OR [3,2,1] for 24 BIT OR [4,3,2,1] for 32 BIT PRODUCT IN [3,2,1] FOR 16 BIT (PRO OR [4,3,2,1] FOR 24 BIT OR [5,4,3,2,1] FOR 32 BI LD CNTR,#17 ID B,#3 ID B,#3 ID A,[B] RC JP M8XXLP IFBNE JP M8XXT RC LD A,[B] IFNC JP M8XXLP S ADC A,[B] ID B,#3 ID JP M8XXL

		RUCTION CYCLES A RUCTION CYCLES M	
		CAND IN [2,1,0]	(ICAND)
		ER IN [3] IN [6,5,4,3]	(IER) (PROD)
(PY248:	LD RC	CNTR,#9	; LD CNTR WITH LENGTH OF : MULTIPLIER FIELD + 1
	LD	B,#6	, MODIFICIER FIELD + 1
	LD	[B-],#0	; CLEAR THREE
	LD	[B-],#0	
	LD	[B-],#0	; UPPER BYTES ; OF PRODUCT
	JP	MP248S	; JUMP TO START
1248LP:	RRC	A	; RIGHT SHIFT HIGH
	X	A,[B-]	; ORDER PRODUCT BYTE
	LD	A,[B]	, onder modeler bird
	RRC	A	; RIGHT SHIFT NEXT LOWER
	X	A,[B-]	; ORDER PRODUCT BYTE
	LD	A,[B]	,
	RRC	A	; RIGHT SHIFT NEXT LOWER
	Х	A,[B-]	ORDER PRODUCT BYTE
/P2485:	LD	A,[B]	,
	RRC	A	; RIGHT SHIFT LOW ORDER
	X	A,[B]	; PRODUCT/MULTIPLIER
	IFNC	,[2]	; TEST LOW ORDER
	JP	MP248T	; MULTIPLIER BIT
	RC		,
	LD	B,#0	: LOAD ACC WITH LOW ORDER
	LD	A,[B]	MULTIPLICAND BYTE
	LD	B,#4	<pre>: LOAD ACC WITH LOW ORDER ; MULTIPLICAND BYTE ; ADD LOW ORDER ICAND ; BYTE TO NEXT TO LOW ; ORDER PRODUCT BYTE ; LOAD ACC WTIH MIDDLE ; MULTIPLICAND BYTE ; ADD MIDDLE ICAND BYTE ; TO NEXT TO HIGH ORDER ; MULTIPLICAND BYTE ; LOAD ACC WITH HIGH ORDER ; MULTIPLICAND BYTE ; ADD HIGH ORDER ICAND BYTE ; TO HIGH ORDER PROD BYTE ; DECREMENT CNTR AND JUMP</pre>
	ADC	A,[B]	BYTE TO NEXT TO LOW
	Х	A,[B]	ORDER PRODUCT BYTE
	LD	B,#1	: LOAD ACC WTIH MIDDLE
	LD	A,[B]	MULTIPLICAND BYTE
	LD	B,#5	; ADD MIDDLE ICAND BYTE
	ADC	A,[B]	TO NEXT TO HIGH ORDER
	Х	A,[B]	MULTIPLICAND BYTE
	LD	B,#2	; LOAD ACC WITH HIGH ORDER
	LD	A,[B]	; MULTIPLICAND BYTE
	LD	B,#6	; ADD HIGH ORDER ICAND BYTE
	ADC	A,[B]	; TO HIGH ORDER PROD BYTE
	DRSZ	CNTR	; TO HIGH ORDER PROD BYTE ; DECREMENT CNTR AND JUMP ; BACK TO LOOF; CNTR
	JP	M248LP	; BACK TO LOOP; CNTR
			; CANNOT EQUAL ZERO
/P248T:	LD	B,#6	; HIGH ORDER PRODUCT
	LD	A,[B]	; BYTE TO ACC
	DRSZ	CNTR	; DECREMENT AND TEST
	JMP	M248LP	; CNTR FOR ZERO
	RET		; RETURN FROM SUBROUTINE

MX1616—FAST 16 BY 16 MULTIPLICATION SUBROUTINE 39 BYTES 498 INSTRUCTION CYCLES AVERAGE 546 INSTRUCTION CYCLES AVERAGE MULTIPLICAND IN [1,0] (ICAND) MULTIPLIER IN [3,2] (IER) PRODUCT IN [5,4,3,2] (PROD) ; LD CNTR WITH LENGTH OF MX1616: CNTR,#17 LD RC MULTIPLIER FIELD + 1 ; ЪD B,#5 ЪD [B**—**],#0 ; CLEAR UPPER TWO [B—],#0 LD PRODUCT BYTES ; PRODUCT BY: ; JUMP TO START MXSTRT JP ; RIGHT SHIFT MX1616L: RRC Α Х A,[B-] UPPER PRODUCT BYTE ; LD A,[B] ; RIGHT SHIFT NEXT LOWER RRC Α A,[B-] PRODUCT BYTE Х ; MXSTRT: T₁D A,[B] ; RIGHT SHIFT PRODUCT RRC Α Х A,[B-] UPPER MULTIPLIER BYTE ; LD A,[B] ; RIGHT SHIFT PRODUCT RRC Α Х A,[B] LOWER MULTIPLIER BYTE ; IFNC TEST LOW ORDER ; JP MX1616T MULTIPLIER BIT ; RC ; LOAD ACC WITH LOWER LD B,#0 MULTIPLICAND BYTE LD A,[B] ; ЪD B,#4 ; ADD LOWER ICAND BYTE ADC TO NEXT TO HIGH A,[B] ; ORDER PRODUCT BYTE Х A,[B] ; ; LOAD ACC WITH UPPER T₁D B,#1 MULTIPLICAND BYTE LD A,[B] ; LD B,#5 ; ADD UPPER ICAND BYTE TO ADC A,[B] ; HIGH ORDER PRODUCT ; DECREMENT CNTR AND JUMP HIGH ORDER PRODUCT DRSZ CNTR JP MX1616LBACK TO LOOP; CNTR ; CANNOT EQUAL ZERO ; MX1616T: LD B,#5 ; HIGH ORDER PRODUCT A,[B] BYTE TO ACC ЪD ; ; DECREMENT AND TEST DRSZ CNTR JP CNTR FOR ZERO MX1616L; RET RETURN FROM SUBROUTINE

RC;MULTIPLIER FIELD +LD $B,\#5$ LD $[B-],\#0$;CLEAR UPPER TWOLD $[B-],\#0$;PRODUCT BYTESAl616X:LDA, $[B]$;FIVE INSTRUCTIONAl616L:RRCA;PROGRAM LOOP TOXA, $[B-]$;RIGHT SHIFTIFBNE#1;PRODUCT/MULTIPLIERJPM1616X;LOOP JUMP BACKCLRA;CLEAR ACCIFNC;TEST LOW ORDERJPM1616T;MULTIPLIER BITRC.LOAD ACC WITH LOWERLDB,#0;LOAD ACC WITH LOWERLDA, $[B]$;ON LOWER ICAND BYTEADCA, $[B]$;ORDER FRODUCT BYTELDB,#4;ADD LOWER ICAND BYTELDB,#1;LOAD ACC WITH UPPERLDB,#1;LOAD ACC WITH UPPERLDA, $[B]$;MULTIPLICAND BYTE	
807 INSTRUCTION CYCLES MAXIMUM] MULTIPLICAND IN [1,0] (ICAND) MULTIPLIER IN [3,2] (IER) PRODUCT IN [5,4,3,2] (PROD) RC LD CNTR,#17 ; LD CNTR WITH LENGTH 0 RC ; MULTIPLIER FIELD + LD B,#5 LD [B-],#0 ; CLEAR UPPER TWO LD [B-],#0 ; FRODUCT BYTES R1616X: LD A,[B] ; FIVE INSTRUCTION I1616L: RRC A X A,[B] ; FIVE INSTRUCTION FOOD X A,[B] ; FIVE INSTRUCTION FOOD X A,[B] ; FRODUCT /MULTIPLIER JP M1616X ; LOOP JUMP BACK CLR A ; CLEAR ACC IFNC ; TEST LOW ORDER JP M1616T ; MULTIPLIER BIT RC ; TEST LOW ORDER LD B,#0 ; LOAD ACC WITH LOWER LD A,[B] ; ORDER FRODUCT BYTE LD B,#4 ; ADD LOWER ICAND BYTE ADC A,[B] ; ORDER FRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD B,#1 ; LOAD ACC WITH UPPER LD B,#1 ; LOAD ACC WITH UPPER LD B,#1 ;	
MULTIPLIER IN $[3,2]$ (IER) PRODUCT IN $[5,4,3,2]$ (IER) PRODUCT IN $[5,4,3,2]$ MP1616:LDCNTR,#17; LD CNTR WITH LENGTH 0 RCRC; MULTIPLIER FIELD + LDB,#5LD[B-],#0; CLEAR UPPER TWO IDDLD[B-],#0; PRODUCT BYTESM1616X:LDA,[B]XA,[B]; FIVE INSTRUCTION YROGRAM LOOP TO XM1616L:RRCAYPM1616X; LOOP JUMP BACK CLRCLRA; CLEAR ACC IFNCJPM1616T; MULTIPLIER BIT RCLDB,#0; LOAD ACC WITH LOWER LDLDB,#4; ADD LOWER ICAND BYTE ADCA,[B]; ONEXT TO LOW XA,[B]XA,[B]; ONEXT TO LOW RCXA,[B]; MULTIPLICAND BYTE LDLDB,#1; LOAD ACC WITH UPPER LDLDB,#1; LOAD ACC WITH UPPER LDLDB,#1; LOAD ACC WITH UPPER LDLDB,#1; LOAD ACC WITH UPPER LDLDB,#1; LOAD ACC WITH UPPER TADCLDB,#5; ADD UPPER ICAND BYTE ADCADCA,[B]; HIGH ORDER PRODUCT DRSZJPM1616L; CNTR EQUAL TO ZERO	
MP1616:LDCNTR,#17;LDCNTR WITH LENGTH ORC;MULTIPLIER FIELD +LDB,#5LD[B-],#0;PRODUCT BYTESM1616X:LDA,[B];FIVE INSTRUCTIONM1616L:RRCA;PROGRAM LOOP TOXA,[B-];RIGHT SHIFTIFBNE#1;PRODUCT/MULTIPLIERJPM1616X;LOOP JUMP BACKCLRA;CLEAR ACCIFNC;TEST LOW ORDERJPM1616T;MULTIPLIER BITRCLDB,#4;ADD LOWER ICAND BYTELDB,#4;ADD LOWER ICAND BYTELDB,#4;ORDER PRODUCT BYTELDB,#1;LOAD ACC WITH LOWERLDB,#4;ADD LOWER ICAND BYTELDB,#1;LOAD ACC WITH UPPERLDB,#1;LOAD ACC WITH UPPERLDB,#1;LOAD ACC WITH UPPERLDA,[B];MULTIFLICAND BYTEM1616T:LDB,#5;ADD UPPER ICAND BYTEM1616T:LDB,#5;ADD UPPER TEAND TESTJPM1616L;CNTR EQUAL TO ZERO	
RC;MULTIPLIER FIELD +LD $B,\#5$;LD $[B-],\#0$;CLEAR UPPER TWO;LD $[B-],\#0$;PRODUCT BYTESM1616X:LDA, $[B]$ FIVE INSTRUCTIONM1616L:RRCA;PROGRAM LOOP TOXA, $[B-]$ IFBNE#1;FRODUCT /MULTIPLIERJPM1616X;LOOP JUMP BACKCLRA;CLEAR ACCIFNC;JPM1616TMULTIPLIER BITRCLDB,#0LDB,#4ADD LOWER ICAND BYTELDB,#4ADCA, [B];ONEXT TO LOWXA, [B];MULTIPLICAND BYTELDB,#1;LOAD ACC WITH UPPERLDA, [B];MULTIPLICAND BYTELDB,#1;LOAD ACC WITH UPPERLDA, [B];MULTIPLICAND BYTEM1616T:LDB,#5;ADD UPPER ICAND BYTEADCA, [B];HIGH ORDER PRODUCTDRSZCNTR;DECREMENT AND TESTJPM1616L;CNTR EQUAL TO ZERO	
LD B,#5 LD [B-],#0 ; CLEAR UPPER TWO LD [B-],#0 ; PRODUCT BYTES M1616X: LD A,[B] ; FIVE INSTRUCTION M1616L: RRC A ; PROGRAM LOOP TO X A,[B-] ; RIGHT SHIFT IFBNE #1 ; PRODUCT/MULTIPLIER JP M1616X ; LOOP JUMP BACK CLR A ; CLEAR ACC IFNC ; TEST LOW ORDER JP M1616T ; MULTIPLIER BIT RC LD B,#0 ; LOAD ACC WITH LOWER LD A,[B] ; MULTIPLICAND BYTE LD B,#4 ; ADD LOWER ICAND BYTE LD B,#4 ; ADD LOWER TO LOW X A,[B] ; ORDER PRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE LD B,#1 ; LOAD ACC WITH UPPER M1616T: LD B,#5 ; ADD UPPER ICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT M1616T: LD B,#5 ; ADD UPPER ICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT M1616T: LD B,#5 ; ADD UPPER ICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECEBMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	OF
LD $[B-],\#0$; CLEAR UPPER TWO LD $[B-],\#0$; PRODUCT BYTES M1616X: LD A, [B]; FIVE INSTRUCTION M1616L: RRC A ; PROGRAM LOOP TO X A, [B-]; RIGHT SHIFT IFBNE #1 ; PRODUCT/MULTIPLIER JP M1616X ; LOOP JUMP BACK CLR A ; CLEAR ACC IFNC ; TEST LOW ORDER JP M1616T ; MULTIPLIER BIT RC LD B,#0 ; LOAD ACC WITH LOWER LD B,#0 ; LOAD ACC WITH LOWER LD B,#4 ; ADD LOWER ICAND BYTE ADC A, [B] ; ORDEX FODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER M1616T; LD B,#1 ; LOAD ACC WITH UPPER LD A, [B] ; MULTIPLICAND BYTE ADC A, [B] ; MULTIPLICAND BYTE JP M1616T; LD B,#5 ; ADD UPPER ICAND BYTE ADC A, [B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	+ 1
LD $[B-],\#0$; PRODUCT BYTES M1616X: LD A, [B]; FIVE INSTRUCTION M1616L: RRC A ; PROGRAM LOOP TO X A, [B-]; RIGHT SHIFT IFBNE #1 ; PRODUCT/MULTIPLIER JP M1616X ; LOOP JUMP BACK CLR A ; CLEAR ACC IFNC ; TEST LOW ORDER JP M1616T ; MULTIPLIER BIT RC LD B,#0 ; LOAD ACC WITH LOWER LD A, [B] ; MULTIPLICAND BYTE LD B,#4 ; ADD LOWER ICAND BYTE ADC A, [B] ; ORDER PRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A, [B] ; MULTIPLICAND BYTE M1616T: LD B,#1 ; LOAD ACC WITH UPPER LD A, [B] ; MULTIPLICAND BYTE M1616T: LD B,#5 ; ADD UPPER ICAND BYTE ADC A, [B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	
MIGIGL: RRC A ; PROGRAM LOOP TO X A,[B-] ; RIGHT SHIFT IFBNE #1 ; PRODUCT/MULTIPLIER JP MIGIGX ; LOOP JUMP BACK CLR A ; CLEAR ACC IFNC ; TEST LOW ORDER JP MIGIGT ; MULTIPLIER BIT RC LD B,#0 ; LOAD ACC WITH LOWER LD A,[B] ; MULTIPLICAND BYTE LD B,#4 ; ADD LOWER ICAND BYTE ADC A,[B] ; ORDER PRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; MULTIPLICAND BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP MIGIGL ; CNTR EQUAL TO ZERO	
X A,[B-]; RIGHT SHIFT IFBNE #1; PRODUCT/MULTIPLIER JP M1616X; LOOP JUMP BACK CLR A ; CLEAR ACC IFNC ; TEST LOW ORDER JP M1616T; MULTIPLIER BIT RC LD B,#0 ; LOAD ACC WITH LOWER LD A,[B]; MULTIPLICAND BYTE LD B,#4 ; ADD LOWER ICAND BYTE ADC A,[B]; ORDER PRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT M1616T: LD B,#5 ; ADD UPPER ICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
IFBNE #1 ; PRODUCT/MULTIPLIER JP M1616X ; LOOP JUMP BACK CLR A ; CLEAR ACC IFNC ; TEST LOW ORDER JP M1616T ; MULTIPLIER BIT RC	
IFBNE#1;PRODUCT/MULTIPLIERJPM1616X;LOOP JUMP BACKCLRA;CLEAR ACCIFNC;TEST LOW ORDERJPM1616T;MULTIPLIER BITRC.LOAD ACC WITH LOWERLDA,[B];MULTIPLICAND BYTELDB,#0;LOAD ACC WITH LOWERLDA,[B];MULTIPLICAND BYTELDB,#4;ADD LOWER ICAND BYTEADCA,[B];ORDER FRODUCT BYTELDB,#1;LOAD ACC WITH UPPERLDA,[B];MULTIPLICAND BYTEADCA,[B];MULTIPLICAND BYTEADCA,[B];HIGH ORDER FRODUCTM1616T:LDB,#5;ADD UPPER ICAND BYTEADCA,[B];HIGH ORDER FRODUCTDRSZCNTR;DECREMENT AND TESTJPM1616L;CNTR EQUAL TO ZERO	
JP M1616X ; LOOP JUMP BACK CLR A ; CLEAR ACC IFNC ; TEST LOW ORDER JP M1616T ; MULTIPLIER BIT RC LD B,#0 ; LOAD ACC WITH LOWER LD A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; TO NEXT TO LOW X A,[B] ; ORDER FRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE M1616T: LD B,#1 ; LOAD ACC WITH UPPER ADC A,[B] ; MULTIPLICAND BYTE M1616T: LD B,#5 ; ADD UPPER ICAND BYTE ADC A,[B] ; HIGH ORDER FRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	ER.
CLR A ; CLEAR ACC IFNC ; TEST LOW ORDER JP M1616T ; MULTIPLIER BIT RC LD B,#0 ; LOAD ACC WITH LOWER LD A,[B] ; MULTIPLICAND BYTE LD B,#4 ; ADD LOWER ICAND BYTE ADC A,[B] ; TO NEXT TO LOW X A,[B] ; ORDER PRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE M1616T: LD B,#5 ; ADD UPPER ICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
IFNC ; TEST LOW ORDER JP M1616T ; MULTIPLIER BIT RC LD B,#0 ; LOAD ACC WITH LOWER LD A,[B] ; MULTIPLICAND BYTE LD B,#4 ; ADD LOWER ICAND BYTE ADC A,[B] ; TO NEXT TO LOW X A,[B] ; ORDER PRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; HIGH ORDER FRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
JP M1616T ; MULTIPLIER BIT RC LD B,#0 ; LOAD ACC WITH LOWER LD A,[B] ; MULTIPLICAND BYTE LD B,#4 ; ADD LOWER ICAND BYTE ADC A,[B] ; TO NEXT TO LOW X A,[B] ; ORDER PRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
RC LD B,#0 ; LOAD ACC WITH LOWER LD A,[B] ; MULTIFLICAND BYTE LD B,#4 ; ADD LOWER ICAND BYTE ADC A,[B] ; TO NEXT TO LOW X A,[B] ; ORDER FRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIFLICAND BYTE ADC A,[B] ; MULTIFLICAND BYTE ADC A,[B] ; HIGH ORDER FRODUCT DRSZ CNTR ; DECREMENT AND TEST JP MIG16L ; CNTR EQUAL TO ZERO	
LD B,#0 ; LOAD ACC WITH LOWER LD A,[B] ; MULTIPLICAND BYTE LD B,#4 ; ADD LOWER ICAND BYTE ADC A,[B] ; TO NEXT TO LOW X A,[B] ; ORDER FRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; HIGH ORDER FRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
LD A,[B] ; MULTIPLICAND BYTE LD B,#4 ; ADD LOWER ICAND BYTE ADC A,[B] ; TO NEXT TO LOW X A,[B] ; ORDER FRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; HIGH ORDER FRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
LD B,#4 ; ADD LOWER ICAND BYTE ADC A,[B] ; TO NEXT TO LOW X A,[B] ; ORDER FRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE ADC A,[B] ; HIGH ORDER FRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
ADC A,[B] ; TO NEXT TO LOW X A,[B] ; ORDER PRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE M1616T: LD B,#5 ; ADD UPPER ICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
X A,[B] ; ORDER PRODUCT BYTE LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIFLICAND BYTE M1616T: LD B,#5 ; ADD UPPER ICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	-
LD B,#1 ; LOAD ACC WITH UPPER LD A,[B] ; MULTIPLICAND BYTE M1616T: LD B,#5 ; ADD UPPER ICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	гE
LD A,[B] ; MULTIPLICAND BYTE M1616T: LD B,#5 ; ADD UPPER ICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
M1616T: LD B,#5 ; ADD UPPER ICAND BYTE ADC A,[B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
ADC A,[B] ; HIGH ORDER PRODUCT DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
DRSZ CNTR ; DECREMENT AND TEST JP M1616L ; CNTR EQUAL TO ZERO	
JP M1616L ; CNTR EQUAL TO ZERO	ت د
KET ; KETUKN FKOM SUBROUTIN	
	INE

28 BYTES 861 (OR 14 1029 (OR 1 EXTENDABLE PARAMETE	725,1473) INST ROUTINE FOR M RS, WITH NUMBH	T. CYCLES AVERAGE T. CYCLES MAXIMUM MY16XX BY CHANGING
OR	[6,5,4,3,2]]	R 16 BIT (PROD) FOR 24 BIT
LD	CNTR,#17	; LD CNTR WITH LENGTH OF ; MULTIPLIER FIELD + 1 ; #17 FOR MY1616 ; (#25 FOR MY1624) ; (#33 FOR MY1632)
LD	B,#5	; #5 FOR MY1616 ; (#6 FOR MY1624) ; (#7 FOR MY1632)
LD LD RC	[B-],#0 [B-],#0	: CLEAR UPPER TWO ; PRODUCT BYTES
LD RRC X IFBNE JP IFNC	A,[B] A A,[B-] #1 M16XS	; FIVE INSTRUCTION ; PROGRAM LOOP TO ; RIGHT SHIFT ; PRODUCT/MULTIPLIER ; LOOP JUMP BACK ; TEST LOW ORDER
JP RC LD	MY16XT B,#4	; MULTIPLIER BIT ; #4 FOR MY1616 ; (#5 FOR MY1624)
LD LD ADC X IFBNE	X,#0 A,[X+] A,[B] A,[B+] #2	; (#6 FOR MY1632) ; LOAD ACC WITH ; MULTIFLICAND BYTES ; ADD MULTIFLICAND TO ; HI TWO PROD. BYTES ; LOOP BACK FOR SECOND
TD TD	MY16XL B,#5	; MULTIFLICAND BYTE ; #5 FOR MY1616 ; (#6 FOR MY1624) ; (#7 FOR MY1632)
DRSZ JP RET	CNTR MY16XS	; DECREMENT AND TEST ; CNTR EQUAL TO ZERO ; RETURN FROM INTERRUPT
	28 BYTES 861 (OR 14 1029 (OR 1 EXTENDABLE PARAMETE REMAININ MULTIPLICA MULTIPLICA PRODUCT IN OR LD LD LD LD LD LD LD RC LD LD RC LD LD LD LD LD LD LD LD LD LD	28 BYTES 861 (OR 1473, 2213) INS 1029 (OR 1725,1473) INS EXTENDABLE ROUTINE FOR PARAMETERS, WITH NUMB REMAINING A CONSTANT MULTIFLICAND IN [1,0] NULTIFLICAND IN [1,0] PRODUCT IN [5,4,3,2] FO OR [6,5,4,3,2] DRC LD [B-],#0 CD [B-],#0 ICD [B-],#0 ICD [B-],#0 ICD [A,[B] RCC A JP M16XS IFNC JP JP M16XS IFBNE

		(MY2448)		
24 B	Y 48 MULTIPL 28 BYTES	ICATION SUBRO	JTINE	
		CODE, MINIMAL	RAM	
		TRUCTION CYCL		
MULT	5457 INS IPLICAND IN	TRUCTION CYCL	LS MAXIMUM (ICAND)	
MULT	IPLIER IN [8	,7,6,5,4,3]	(IER)	
PROD	UCT IN [11,1	0,9,8,7,6,5,4	,3] (PROD)	
	EXAMPLE: (MY			
48 B	Y 24 MULTIPL 28 BYTES	ICATION SUBRO	JTINE	
		CODE, NON MIN	IMAL RAM	
		TRUCTION CYCL		
MULT		TRUCTION CYCL [5,4,3,2,1,0]		
	IPLIER IN [8	• • • • • • •	(IER)	
PROD	UCT IN [14,1	3,12,11,10,9,	3,7,6] (PROD)	
Y2448:	; (OR MY48	24)		
	LD		; LD CNTR WITH LENGTH O	F
			; MULTIPLIER FIELD +	- 1
			; #49 FOR MY2448 ; (#25 FOR MY4824)	
	LD	B,#11	; TOP OF PROD TO B PTR	
			; #11 FOR MY2448	
LRLUP:	LD	[B —],#O	; (#14 FOR MY4824) ; CLR UNTIL TOP OF IER	
JANDOF :	IFBNE	[B-],#0 #8	; #8 FOR BOTH MY2448	
	JP	CLRLUP	; AND MY4824	
	RC	A [D]	; INITIALIZE CARRY	
SHFTLP:	LD ADC		; RIGHT SHIFT PRODUCT ; AND MULTIPLIER	
	X	A,[B-]	; UNTIL TOP OF ICAND	
	IFBNE	#2	; #2 FOR MY2448	
	JP IFNC		; (#5 FOR MY4824) ; TEST LOW ORDER	
	JP		; MULTIPLIER BIT	
	LD		; TOP OF IER + 1 TO B P	TR
	LD	X,#O	; START OF ICAND TO X P	TR
DDLUP:	RC LD	A [X+]	; ADD MULTIPLICAND TO T	0P
	ADC	A,[B]		
	Х	A,[B+]	; MULTIPLIER UNTIL T	OP
	IFBNE	#12	; OF PRODUCT + 1	
	JP	ADDLUP	; #12 FOR MY2448 ; (#15 FOR MY4824)	
YTEST:	LD	B,#11	; TOP OF PROD TO B PTR	
			; #11 FOR MY2448	
	DRSZ	CNTR	; (#14 FOR MY4824) : DECREMENT AND TEST	
	JP	SHFTLP	; CNTR FOR ZERO	
			; RETURN FROM SUBROUTIN	Έ
	RET		, REFORM FROM DODROOTIN	-

3.0 DIVISION

The COP 800 divisions are all based on shifting the dividend left up into a test field equal in length to the number of bytes in the divisor. The divisor is resident immediately above this test field. After each shift cycle of the dividend into the test field, a trial subtraction is made of the test field minus the divisor. If the divisor is found equal to or less than the contents of the test field, then the divisor is subtracted from the test field and a 1's quotient digit is recorded by setting the low order bit of the dividend field. The dividend and test field left shift cycle is then repeated. The number of left shift cycles is equal to the number of bit positions in the dividend. The quotient from the division is formed in the dividend field, while the remainder from the division is resident in the test field.

Note that an M byte dividend divided by an N byte divisor will result in an M byte quotient and an N byte remainder. These division algorithms will use M + 2N + 1 bytes of RAM memory space, since the test field is equal to the length of the divisor. The one extra byte is necessary for the shift counter CNTR.

In special cases where the dividend has an upper bound and the divisor has a lower bound, the upper bytes of the dividend may be used as the test field. One example is shown (DV2815), where a 28 bit dividend is divided by a 15-bit divisor. The dividend is less than 2**28 (upper nibble of high order byte is zero), while the divisor is greater than 2**12 (4096) and less than 2**15 (32768). In this case, the upper limit for the quotient is 2**28/2**12, which indicates а qı th th in

upper limit for the quotient is 2**28/2 a 16-bit quotient (2**16) and a 15- quently, the upper two bytes of the div the test field for the remainder, since than the test field (upper two bytes of initially. The minimal code (40 byte) general shown with the example DV3224, w dividend by a 24 bit divisor.	bit remainder. Conse- ridend may be used as the divisor is greater D of the 28-bit dividend) division subroutine is	X1616	 Extendable Houtine for DVXX16 by Changing Parameters, with Number of Bytes (39) Remaining a Constant Fast 16 by 16 Division Subroutine 53 Bytes 638 Instruction Cycles Average 678 Instruction Cycles Maximum Fast 28 by 15 Division Subroutine, Where the Dividend is Less Than 2**28
DIV88 — 8 by 8 Division Subr — 24 Bytes — 201 Instruction Cycl — 209 Instruction Cycl Minimum code DV88 — Fast 8 by 8 Division — 28 Bytes — 194 Instruction Cycl — 202 Instruction Cycl — 202 Instruction Cycl — 131 Bytes — 146 Instruction Cycl — 159 Instruction Cycl DIV168 (or DIV248, DIV328) — 16 (or 24, 32) by 8 I — 26 Bytes — 649 (or 1161, 1801) Cycles Average — 681 (or 1209,1865) Cycles Maximum — Minimum Code — Extendable Routine Changing Parameters of Bytes (26) Remain	es Average es Maximum Subroutine es Average es Maximum /ision Subroutine es Average es Maximum Division Subroutine Instruction Instruction for DIVXX8 by s, with Number	X3216 linimal Ge ytes in Div	and the Divisor is Greater than 2**12 (4096) and Less than 2**15 (32768) 43 Bytes 640 Instruction Cycles Average 696 Instruction Cycles Maximum Fast 32 by 16 Division Subroutine 70 Bytes 1591 Instruction Cycles Average 1591 Instruction Cycles Maximum neral Division Subroutine for any Number of idend and Divisor 40 Bytes Minimal Code DV3224 Used as Example, with 3879 Instruction Cycles Maximum

FDV168

FDV248

FDV328

- Fast 16 by 8 Division Subroutine

- 481 Instruction Cycles Average

- 490 Instruction Cycles Maximum

- Fast 24 by 8 Division Subroutine

- 813 Instruction Cycles Average

- 826 Instruction Cycles Maximum

- Fast 32 by 8 Division Subroutine

- 1209 Instruction Cycles Average

- 979 Instruction Cycles Average

- 1067 Instruction Cycles Maximum

— 24 (or 32) by 16 Division Subroutine
— 39 Bytes

- 1694 (or 2410) Inst. Cycles Average

- Extendable Routine for DVXX16 by

- 1886 (or 2766) Inst. Cycles Maximum

- 1226 Instructions Maximum

- 35 Bytes

- 38 Bytes

- 42 Bytes

DV1616 — 16 by 16 Division Subroutine — 34 Bytes

- Minimum Code

- Minimum code

Divide by 16 Subroutines:

DV2416 (or DV3216)

DIV88-8 BY 8 DIVISION SUBROUTINE

	51 8 DIVISION 30	DROUTINE	
	MINIMUM CODE	3	
	24 BYTES		
	201 INSTRUCI	ION CYCLES	AVERAGE
	209 INSTRUCI	ION CYCLES	MAXIMUM
	DIVIDEND		(DD)
	DIVISOR IN [[2]	(DR)
	QUOTIENT IN	[0]	(QUOT)
	REMAINDER IN	[1]	(TEST FIELD)
DIV88:	LD		
DIV66:	LD D	B,#1	; LOAD CNTR WITH LENGTH : OF DIVIDEND FIELD
	LD	ь,#⊥ [B],#O	; OF DIVIDEND FIELD : CLEAR TEST FIELD
DIV88S	RC	[b],#0	; CLEAR IESI FIELD
DIVOOD	LD	B,#0	
	-	A,[B]	
	ADC	A,[B]	: LEFT SHIFT DIVIDEND
	X	A,[B+]	, DEFI SHIFI DIVIDEND
	LD	A,[B]	
	-		; LEFT SHIFT TEST FIELD
	X	A,[B]	,
	LD	A,[B+]	; TEST FIELD TO ACC
	SC	,[]	: TEST SUBTRACT DIVISOR
	SUBC	A,[B]	FROM TEST FIELD
	IFNC	, L _	; TEST IF BORROW
	JP	DIV88B	FROM SUBTRACTION
	LD	B,#1	
	Х	A,[B—]	; TO TEST FIELD
	SBIT	0,[B]	; SET QUOTIENT BIT
DIV88B:	DRSZ	CNTR	: DECREMENT AND TEST
	JP	DIV88S	CNTR FOR ZERO
	RET		; RETURN FROM SUBROUTINE

		UCTION CYCLES	
	159 INSTR	UCTION CYCLES	MAXIMUM
	DIVIDEND	IN [0]	(DD)
	DIVISOR I	N [2]	(DR)
	QUOTIENT		(QUOT)
	REMAINDER	IN [1]	(TEST FIELD)
TDV88:	LD	B,#1	
	LD	[B—],#O	; CLEAR TEST FIELD
	RC	4 551	
	LD	A,[B]	
	ADC	A,[B]	; LEFT SHIFT DIVIDEND
	X	A,[B+]	
	LD ADC	A,[B]	: LEFT SHIFT TEST FIELD
	X	A,[B] A,[B]	; DEFI SHIFI IESI FIEDD
	LD	A,[B]	; TEST FIELD TO ACC
	SC	A,[DT]	; TEST SUBTRACT DIVISOR
	SUBC	A,[B]	; FROM TEST FIELD
	IFNC	,[2]	; TEST IF BORROW
	JP	DVBP1	; FROM SUBTRACTION
	LD	B,#1	; SUBTRACTION RESULT
	X	A,[B-]	; TO TEST FIELD
	SBIT	0,[B]	; SET QUOTIENT BIT
	RC		
OVBP1:	LD	В,#О	; THIS 16 BYTE SECTION
	LD	A,[B]	; OF PROGRAM CODE
	ADC	A,[B]	; CONTAINS
	Х	A,[B+]	; 16 INSTRUCTIONS,
	LD	A,[B]	; AND REPRESENTS THE
	ADC	A,[B]	; PROCESSING FOR THE
	Х	A,[B]	; GENERATION OF
	LD	A,[B+]	; 1 QUOTIENT BIT.
	SC		;
	SUBC	A,[B]	; THE PROGRAM CODE
	IFNC		; EXECUTION TIMES IS 16
	JP	DVBP2	; INSTRUCTION CYCLES
	LD	B,#1	; FOR A O'S QUOTIENT BIT
	Х	A,[B-]	; AND 19 INSTRUCTION
	SBIT	0,[B]	; CYCLES FOR A 1'S
	RC		; QUOTIENT BIT.
;			;
OVBP2:	LD	B,#0	; REPEAT THE ABOVE
,			;
;DVBP3:			CECTION OF CODE FILE
, DUPD4 .			;SECTION OF CODE FIVE
;DVBP4:			;MORE TIMES FOR A
; ;DVBP5:			;MORE TIMES FOR A
,DVD10.			;TOTAL OF SIX TIMES
; DVBP6:			;IOIAL OF SIX IIMES
-			;
;			,
, DVBP7:	LD	В,#О	
	LD	A,[B]	
	ADC	A,[B]	; LEFT SHIFT DIVIDEND
	Х	A,[B+]	
	LD	A,[B]	
	ADC	A,[B]	; LEFT SHIFT TEST FIELD
	Х	A,[B]	
	LD	A,[B+]	; TEST FIELD TO ACC
	SC		; TEST SUBTRACT DIVISOR
	SUBC	A,[B]	; FROM TEST FIELD
	IFNC		; TEST BORROW FROM SUBC
	RET		; RETURN FROM SUBROUTINE
	LD	B,#1	; SUBTRACTION RESULT
	Х	A,[B-]	; TO TEST FIELD
	SBIT	0,[B]	; SET QUOTIENT BIT
	RET		; RETURN FROM SUBROUTINE

	MINIMUM	CODE			
	26 BYTES				
			INST. CYCLES AVER INST. CYCLES MAXI		
			FOR DIVXX8 BY CHA		
			MBER OF BYTES (26		
		G A CONSTAN		,	
	DIVIDEND	IN [1,0] F	OR 16 BIT	(DD)	
			FOR 24 BIT		
			0] FOR 32 BIT		
	DIVISOR	IN [3] FOR	16 BIT	(DR)	
		OR [4] FOR			
		OR [5] FOR		(0100)	
	QUOTIENT	IN [1,0] F	FOR 24 BIT	(QUOT)	
			0] FOR 32 BIT		
	REMAINDE	R IN [2] FO		(TEST FIELD)	
		OR [3] FOR		()	
		OR [4] FOR			
DIV168:	LD	CNTR,#16	; LOAD CNTR WI	TH LENGTH	
			; OF DIVIDE		
			; #16 FOR DIV1	68	
			; (#24 FOR DIV		
	TD	P #0	; (#32 FOR DIV		
	LD	B,#2	; (#3 FOR DIV1		
			; (#3 FOR DIV2 ; (#4 FOR DIV3		
	LD	[B],#O	; CLEAR TEST F		
DVXX8L:	RC		, -		
	LD	B,#0			
DXX8LP:	LD	A,[B]	; LEFT SHIFT D		
	ADC	A,[B]	; AND TEST	FIELD	
	X	A,[B+]	. #7 EOD DIVIS	0	
	IFBNE JP	#3 DXX8LP	; #3 FOR DIV16 ; (#4 FOR DIV2		
	01	DAAGEI	; (#4 FOR DIV2 ; (#5 FOR DIV3		
	LD	A,[B—]	; DIVISOR TO A		
	IFC		; TEST IF BIT		
	JP	DVXX8S	; OF TEST F	IELD***	
	IFGT	A,[B]	; TEST DIVISOR		
	JP	DVXX8T	; THAN REMA	INDER	
WVV00-	SC	A FD7	DEMATNDED TO	400	
DVXX8S:	X SUBC	A,[B] A,[B]	; REMAINDER TO ; SUBTRACT DIV		
	X	A,[B]	; FROM REMA		
	LD	B,#0	, 1100 1000		
	SBIT	0,[B]	; SET QUOTIENT	BIT	
DVXX8T:	DRSZ	CNTR	; DECREMENT AN		
	JP	DVXX8L	; CNTR FOR		
	RET		; RETURN FROM	SUBROUTINE	
;					
;					
***	SPECIAL	CASE FOR DI	VISION WHERE NUME	ER OF BYTES	
;				F BYTES IN DIVISOR, AND	
;				T. THE SHIFTED DIVIDEND	
;				THE TEST FIELD AND	
;				HAT NO SUBTRACTION	
	UCCURS.	IN INIS CAS.	е ч т.Э ріт мірр	BE SHIFTED OUT OF	
; ;		FIELD AND	AN OVERRIDE CHETE	ACTION MUST BE PERFORMED	

FDV168—FAST 16 BY 8 DIVISION SUBROUTINE

121100	35 BYTES	VISION SUBROUTINE	-	
		TION CYCLES AVER TION CYCLES MAXI		
	DIVIDEND IN DIVISOR IN QUOTIENT IN REMAINDER I	[3] [1,0]	(D (Q	D) R) UOT) EST FIELD)
FDV168:	LD LD LD LD	CNTR,#16 B,#3 [B],#0 B,#0	;;;	LOAD CNTR WITH LENGTH OF DIVIDEND FIELD CLEAR TEST FIELD
FD1681:	RC LD ADC X LD	A,[B] A,[B] A,[B+] A,[B]	;	LEFT SHIFT DIVIDEND LO
	ADC X LD	A,[B] A,[B+] A,[B]	;	LEFT SHIFT DIVIDEND HI
	ADC X LD	A,[B] A,[B] A,[B+]	;	
	IFC JP SC	FD168B	; ; ;	OF TEST FIELD*** TEST SUBTRACT DIVISOR
	SUBC IFNC JP	A,[B] FD168T	;;;	FROM TEST FIELD TEST IF BORROW FROM SUBTRACTION
FD168R:	LD X LD	B,#2 A,[B] B,#0	; ;	SUBTRACTION RESULT TO TEST FIELD
	SBIT DRSZ JP RET	O,[B] CNTR FD168L	;;;;;	
FD168T:	DRSZ JP RET	CNTR FD168S	;;;	DECREMENT AND TEST CNTR FOR ZERO RETURN FROM SUBROUTINE
FD168B:	SUBC JP	A,[B] FD168R	; ;	SUBTRACT DIVISOR FROM TEST FIELD***

	813 INSTRUCTIO 826 INSTRUCTIO			
	DIVIDEND IN [2 DIVISOR IN [4] QUOTIENT IN [2 REMAINDER IN]		(D	R)
FDV248:	LD (LD H	CNTR,#24 3.#4	;	LOAD CNTR WITH LENGTH OF DIVIDEND FIELD CLEAR TEST FIELD
	LD	B1.#0	:	CLEAR TEST FIELD
FD2485:	LD H	3,#0	,	
FD248L:	RC			
	LD A	A,[B]		
			;	LEFT SHIFT DIVIDEND LO
		A,[B+]		
	LD A	A,[B]		
	ADC A	A,[B]	;	LEFT SHIFT DIVIDEND MID
	X A	A,[B+]		
	LD A	A,[B]		
	ADC A	A,[B]	;	LEFT SHIFT DIVIDEND HI
	X A	A,[B+]		
		A,[B]		
		A,[B]	;	LEFT SHIFT TEST FIELD
		A,[B]		
	LD A	A,[B+]		
	IFC		;	TEST IF BIT SHIFTED OUT
		D248B	;	OF TEST FIELD ***
	SC			TEST SUBTRACT DIVISOR
	SUBC A			FROM TEST FIELD
	IFNC			TEST IF BORROW
	JP I	D248T	;	FROM SUBTRACTION
FD248R:		3,#3	;	SUBTRACTION RESULT
	X I	A,[B]	;	TO TEST FIELD
	LD H	3,#0		
	SBIT (),[B]	;	SET QUOTIENT BIT DECREMENT AND TEST
	DRSZ (NTR	;	DECREMENT AND TEST CNTR FOR ZERO
		D248L	;	UNTE FOR ZERO
	RET			RETURN FROM SUBROUTINE
FD248T:				DECREMENT AND TEST
			;	
PDOADD.	RET		;	RETURN FROM SUBROUTINE
FD248B:		A,[B] 7D248R	;	SUBTRACT DIVISOR FROM TEST FIELD ***
	JF 1	DZ48K	;	TPJT 4,1FTD 444

979 (OR 1655,2459) INSTRUCTION CYCLES AVERAGE 1067 (OR 1787,2635) INSTRUCTION CYCLES MAXIMUM DIVIDEND IN [1,0] (DD) DIVISOR IN [5,4] (DR) QUOTIENT IN [1,0] (QUOT) REMAINDER IN [3,2] (TEST FIELD) 1616: LD CNTR,#16 : LOAD CNTR WITH LENGTH ; OF DIVIDEND FIELD LD B,#3 LD [B-],#0 ; CLEAR LD [B],#0 ; TEST FIELD 616S: RC LD X,#2 ; INITIALIZE X POINTER LD B,#0 ; INITIALIZE B POINTER CG16L: LD A,[B] ; LEFT SHIFT DIVIDEND ADC A,[B] ; AND TEST FIELD X A,[B+] IFBNE #4 JP DV616L SC ; RESET BORROW LD A,[X] ; TEST FIELD HI TO ACC SUBC A,[B] ; SUBT DR LO FROM REM LO LD B,#5 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD HI TO ACC LD B,#5 SUBC A,[C] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#5 SUBC A,[C] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#5 SUBC A,[C] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#5 SUBC A,[C] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#5 SUBC A,[C] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#6 SUBC A,[C] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#6 SUBC A,[C] ; SUBT DR LO FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[C] ; SUBT DR LO FROM REM LO X A,[X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[C] ; SUBT DR LO FROM REM LO X A,[X] ; RESULT LO TO REM LO LD B,#0 SBIT 0,[C] ; SET QUOTIENT BIT		MINIMUM COD 34 BYTES	-		
DIVIDEND IN [1,0] (DD) DIVISOR IN [5,4] (DR) QUOTIENT IN [1,0] (QUOT) REMAINDER IN [3,2] (TEST FIELD) Telefe: LD CNTR,#16 ; LOAD CNTR WITH LENGTH ; OF DIVIDEND FIELD LD B,#3 LD [B-],#0 ; CLEAR LD [B],#0 ; TEST FIELD Telefe: LD X,#2 ; INITIALIZE X POINTER LD B,#0 ; INITIALIZE B POINTER LD A,[B] ; LEFT SHIFT DIVIDEND ADC A,[B] ; LEFT SHIFT DIVIDEND ADC A,[B] ; LEFT SHIFT DIVIDEND ADC A,[B] ; SUBT DR LO FROM REM LO LD A,[X] ; TEST FIELD LO TO ACC SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD HI TO ACC LD B,#5 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#5 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#5 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#5 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#5 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#6 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#6 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT DR LO FROM REM LO X A,[X] ; RESULT LO TO REM LO X A,[X] ; RESULT LO TO REM LO ID B,#0 SBIT 0,[B] ; SET QUOTIENT BIT FG16T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO			5,2459) INSTE	RUCTI	ON CYCLES AVERAGE
DIVISOR IN $[5,4]$ (DR) QUOTIENT IN $[1,0]$ (QUOT) REMAINDER IN $[3,2]$ (TEST FIELD) Tellof: LD CNTR,#16 ; LOAD CNTR WITH LENGTH ; OF DIVIDEND FIELD LD B,#3 LD [B-],#0 ; CLEAR LD [B],#0 ; TEST FIELD Tellof: RC LD X,#2 ; INITIALIZE X POINTER LD B,#0 ; INITIALIZE B POINTER LD A,[B] ; LEFT SHIFT DIVIDEND ADC A,[B] ; LEFT SHIFT DIVIDEND ADC A,[B] ; AND TEST FIELD X A,[B+] IFBNE #4 JP DV616L SC ; RESET BORROW LD A,[X] ; TEST FIELD LO TO ACC SUBC A,[B] ; SUBT DR LO FROM REM LO LD B,#5 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#5 SUBC A,[C] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#5 SUBC A,[C] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#5 SUBC A,[C] ; SUBT DR HI FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[C] ; SUBT DR LO FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[C] ; SUBT DR LO FROM REM LO LD B,#4 SUBC A,[C] ; SUBT DR LO FROM REM LO LD B,#4 SUBC A,[C] ; SUBT DR LO FROM REM LO LD B,#4 SUBC A,[C] ; SUBT DR LO FROM REM LO LD B,#0 SBIT 0,[C] ; SET QUOTIENT BIT FG16T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO		1067 (OR 17	87,2635) INSI	FRUCT	ION CYCLES MAXIMUM
QUOTIENT IN [1,0] REMAINDER IN [3,2](QUOT) (TEST FIELD)'1616:LDCNTR,#16: LOAD CNTR WITH LENGTH ; OF DIVIDEND FIELDLDB,#3 LD[B],#0: CLEAR TEST FIELD'1616:LD[B],#0; TEST FIELD'1616:LD[B],#0; TEST FIELD'1616:LD[B],#0; TEST FIELD'1616:LD[B],#0; TEST FIELD'1616:LDX,#2; INITIALIZE X POINTER LDLDB,#0; INITIALIZE B POINTER'16161:LDA,[B]; LEFT SHIFT DIVIDEND ADCADCA,[B]; LEFT SHIFT DIVIDEND ADCA,[C]XA,[B]; SUBT ST FIELD LO TO ACCSUBCA,[B]; SUBT DR LO FROM REM LO LDLDA,[X]; TEST FIELD HI TO ACCLDB,#5; SUBT DR HI FROM REM HI ; TEST IF BORROWJPDV616T; FROM SUBTRACTION XXA,[C-]; SUBT RESULT HI TO REM HI ; TEST FIELD LO TO ACCLDB,#4;SUBCA,[B]; SUBT RESULT HI TO REM HI ; TEST FIELD LO TO ACCLDB,#4;SUBCA,[B]; SUBT DR LO FROM REM LO XXA,[X]; RESULT LO TO REM LO ; SET ; DECREMENT AND TEST'10B,#0; SET QUOTIENT BIT ; DECREMENT AND TEST'21CNTR; DECREMENT AND TEST'21JPDV616S; CNTR FOR ZERO		DIVIDEND IN	[1,0]	(D	D)
REMAINDER IN [3,2](TEST FIELD)T616:LDCNTR,#16; LOAD CNTR WITH LENGTH ; OF DIVIDEND FIELDLDB,#3;OF DIVIDEND FIELDLD[B-],#0; CLEARLD[B],#0; TEST FIELD7616S:RCLDX,#2; INITIALIZE X POINTERLDB,#0; INITIALIZE B POINTERC616L:LDA,[B]XA,[B]; LEFT SHIFT DIVIDENDADCA,[B]; AND TEST FIELDXA,[B+]; TEST FIELD LO TO ACCSUBCA,[X]; TEST FIELD HI TO ACCLDB,#5;SUBCA,[B]; SUBT DR LO FROM REM LOLDA,[X]; TEST IF BORROWJPDV616T; FROM SUBTRACTIONXA,[X]; TEST FIELD LO TO ACCLDB,#5;SUBCA,[B]; SUBT DR HI FROM REM HIIFNC; TEST IF BORROWJPDV616T; FROM SUBTRACTIONXA,[X]; TEST FIELD LO TO ACCLDB,#4SUBCA,[B]; SUBT RESULT HI TO REM HILDB,#4SUBCA,[B]; SUBT DR LO FROM REM LOXA,[C]; RESULT LO TO REM LOLDB,#0; SET QUOTIENT BITC616T:DRS2CNTRJPDV616S; CNTR FOR ZERO		DIVISOR IN	[5,4]	(D	R)
Tield:LDCNTR,#16:LOAD CNTR WITH LENGTH ;LD $B,#3$ LD $[B-],#0$;CLEAR TEST FIELDG16S:RCTEST FIELDG16L:LD $A,[B]$;LFT SHIFT DIVIDEND ADCADC $A,[B]$;LEFT SHIFT DIVIDEND ADCADC $A,[B]$;AND TEST FIELDY $A,[B]$;AND TEST FIELDX $A,[B+]$;TEST FIELD LO TO ACC SUBCLD $A,[X+]$;TEST FIELD LO TO ACC LDLD $A,[X]$;TEST FIELD HI TO ACC LDLD $B,#5$;SUBCSUBC $A,[B]$;SUBT RESULT HI TO REM HI IFNCJPDV616T;FROM SUBTRACTION XX $A,[Z]$;TEST FIELD LO TO ACC LDLD $B,#4$;SUBC $A,[B]$;SUBC $A,[B]$;LD $B,#0$ SBITO,[B]SBIT $O,[B]$;SETQUOTIENT BIT DCREMENT AND TEST JPOV616S;CNTR FOR ZERO		QUOTIENT IN	[1,0]	(Q	UOT)
; OF DIVIDEND FIELD LD B,#3 LD [B-],#0 ; CLEAR LD [B],#0 ; TEST FIELD 616S: RC LD X,#2 ; INITIALIZE X POINTER LD B,#0 ; INITIALIZE B POINTER 616L: LD A,[B] ; LEFT SHIFT DIVIDEND ADC A,[B] ; LEFT SHIFT DIVIDEND ADC A,[B] ; AND TEST FIELD X A,[B+] IFBNE #4 JP DV616L SC ; RESET BORROW LD A,[X+] ; TEST FIELD LO TO ACC SUBC A,[B] ; SUBT DR LO FROM REM LO LD B,#5 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JF DV616T ; FROM SUBTRACTION X A,[X-] ; SUBT RESULT HI TO REM HI IFNC ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT RESULT HI TO REM HI LD A,[X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT DR LO FROM REM LO LD B,#0 SBIT O,[B] ; SET QUOTIENT BIT 616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO		REMAINDER I	N [3,2]	(T	EST FIELD)
; OF DIVIDEND FIELD LD B,#3 LD [B-],#0 ; CLEAR LD [B],#0 ; TEST FIELD 616S: RC LD X,#2 ; INITIALIZE X POINTER LD B,#0 ; INITIALIZE B POINTER 616L: LD A,[B] ; LEFT SHIFT DIVIDEND ADC A,[B] ; LEFT SHIFT DIVIDEND ADC A,[B] ; AND TEST FIELD X A,[B+] IFBNE #4 JP DV616L SC ; RESET BORROW LD A,[X+] ; TEST FIELD LO TO ACC SUBC A,[B] ; SUBT DR LO FROM REM LO LD B,#5 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JF DV616T ; FROM SUBTRACTION X A,[X-] ; SUBT RESULT HI TO REM HI IFNC ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT RESULT HI TO REM HI LD A,[X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT DR LO FROM REM LO LD B,#0 SBIT O,[B] ; SET QUOTIENT BIT 616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO	V1616:	LD	CNTR.#16	:	LOAD CNTR WITH LENGTH
LD $B,\#3$ LD $[B-],\#0$; CLEAR LD $[B],\#0$; TEST FIELD 616S: RC LD $X,\#2$; INITIALIZE X POINTER LD $B,\#0$; INITIALIZE B POINTER LD $B,\#0$; INITIALIZE B POINTER 616L: LD A, [B]; LEFT SHIFT DIVIDEND ADC A, [B]; AND TEST FIELD X A, [B+] IFBNE $\#4$ JF DV616L SC ; RESET BORROW LD A, [X+]; TEST FIELD LO TO ACC SUBC A, [B]; SUBT DR LO FROM REM LO LD B, $\#5$ SUBC A, [B]; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JF DV616T ; FROM SUBTRACTION X A, [X-]; SUBT RESULT HI TO REM HI IFNC ; TEST FIELD LO TO ACC LD B, $\#4$ SUBC A, [B]; SUBT DR LO FROM REM HI IFNC ; TEST FIELD LO TO ACC LD B, $\#4$ SUBC A, [B]; SUBT DR HI FROM REM HI CD B, $\#4$ SUBC A, [B]; SUBT DR LO FROM REM LO LD B, $\#4$ SUBC A, [B]; SUBT DR LO FROM REM LO X A, [X]; RESULT LO TO REM LO X A, [X]; RESULT LO TO REM LO SBIT O, [B]; SET QUOTIENT BIT C616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO		_			
LD $[B],\#0$; TEST FIELD 616S: RC LD X,#2 ; INITIALIZE X POINTER LD B,#0 ; INITIALIZE B POINTER 616L: LD A, [B] ; LEFT SHIFT DIVIDEND ADC A, [B] ; AND TEST FIELD X A, [B+] IFBNE #4 JP DV616L SC ; RESET BORROW LD A, [X+] ; TEST FIELD LO TO ACC SUBC A, [B] ; SUBT DR LO FROM REM LO LD A, [X] ; TEST FIELD HI TO ACC LD B,#5 SUBC A, [B] ; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JP DV616T ; FROM SUBTRACTION X A, [X-] ; SUBT RESULT HI TO REM HI LD A, [X] ; TEST FIELD LO TO ACC LD B,#5 SUBC A, [B] ; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JP DV616T ; FROM SUBTRACTION X A, [X-] ; SUBT RESULT HI TO REM HI LD A, [X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A, [B] ; SUBT DR LO FROM REM LO X A, [X] ; RESULT LO TO REM LO X A, [X] ; RESULT LO TO REM LO SBIT O, [B] ; SET QUOTIENT BIT C616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO		LD	B,#3		
Gel6S:RCLDX,#2;INITIALIZE X POINTERLDB,#0;INITIALIZE B POINTERGel6L:LDA,[B];LEFT SHIFT DIVIDENDADCA,[B];LEFT SHIFT DIVIDENDADCA,[B];AND TEST FIELDXA,[B+];IFBNEJPDV616L;RESET BORROWLDA,[X+];TEST FIELD LO TO ACCSUBCA,[B];SUBT DR LO FROM REM LOLDA,[X];TEST FIELD HI TO ACCLDB,#5;SUBT DR HI FROM REM HIIFNC;TEST IF BORROWJPDV616T;FROM SUBTRACTIONXA,[X];TEST FIELD LO TO ACCLDB,#4;SUBCSUBCA,[B];SUBT DR LO FROM REM LOLDB,#4;SUBCSUBCA,[B];SUBT DR LO FROM REM LOLDB,#0;SET QUOTIENT BITCHG6T:DRS2CNTR;DECREMENT AND TESTJPDV616S;CNTR FOR ZERO		LD	[B —],#O	;	CLEAR
LD X,#2 ; INITIALIZE X POINTER LD B,#0 ; INITIALIZE B POINTER ADC A,[B] ; LEFT SHIFT DIVIDEND ADC A,[B] ; AND TEST FIELD X A,[B+] IFBNE #4 JP DV616L SC ; RESET BORROW LD A,[X+] ; TEST FIELD LO TO ACC SUBC A,[B] ; SUBT DR LO FROM REM LO LD B,#5 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JP DV616T ; FROM SUBTRACTION X A,[X-] ; SUBT RESULT HI TO REM HI IFNC ; TEST FIELD LO TO ACC LD B,#5 SUBC A,[B] ; SUBT RESULT HI TO REM HI IFNC ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT RESULT HI TO REM HI LD A,[X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT DR LO FROM REM LO X A,[X] ; RESULT LO TO REM LO X A,[X] ; RESULT LO TO REM LO X A,[X] ; RESULT LO TO REM LO SBIT O,[B] ; SET QUOTIENT BIT C616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO		LD	[B],#O	;	TEST FIELD
LD B,#0 ; INITIALIZE B POINTER 616L: LD A,[B] ; LEFT SHIFT DIVIDEND ADC A,[B] ; AND TEST FIELD X A,[B+] IFBNE #4 JP DV616L SC ; RESET BORROW LD A,[X+] ; TEST FIELD LO TO ACC SUBC A,[B] ; SUBT DR LO FROM REM LO LD B,#5 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JF DV616T ; FROM SUBTRACTION X A,[X-] ; SUBT RESULT HI TO REM HI LD A,[X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT RESULT HI TO REM HI LD A,[X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT DR LO FROM REM LO X A,[X] ; RESULT LO TO REM LO X A,[X] ; RESULT LO TO REM LO X A,[X] ; RESULT LO TO REM LO SBIT O,[B] ; SET QUOTIENT BIT C616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO	V616S:	RC			
Gel6L:LDA,[B];LEFT SHIFT DIVIDENDADCA,[B];AND TEST FIELDXA,[B+];AND TEST FIELDJFDV616L;;SC;RESET BORROWLDA,[X+];TEST FIELD LO TO ACCSUBCA,[B];SUBT DR LO FROM REM LOLDB,#5;SUBT DR HI FROM REM HIIFNC;TEST FIELD HI TO ACCJPDV616T;FROM SUBTRACTIONXA,[Z];TEST FIELD LO TO ACCLDB,#4;SUBT CA,[B]SUBCA,[B];SUBT DR LO FROM REM LOZA,[X];TEST FIELD LO TO ACCLDB,#4;SUBT CA,[B]SBITO,[B];SET QUOTIENT BITCl66T:DRS2CNTR;DECREMENT AND TESTJPDV616S;CNTR FOR ZERO					
ADC A, $[B]$; AND TEST FIELD X A, $[B+]$ IFBNE #4 JP DV616L SC ; RESET BORROW LD A, $[X+]$; TEST FIELD LO TO ACC SUBC A, $[B]$; SUBT DR LO FROM REM LO LD A, $[X]$; TEST FIELD HI TO ACC LD B, #5 SUBC A, $[B]$; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JP DV616T ; FROM SUBTRACTION X A, $[X-]$; SUBT RESULT HI TO REM HI LD A, $[X]$; TEST FIELD LO TO ACC LD B, #4 SUBC A, $[B]$; SUBT DR LO FROM REM LO X A, $[X]$; TEST FIELD LO TO ACC LD B, #4 SUBC A, $[B]$; SUBT DR LO FROM REM LO X A, $[X]$; RESULT LO TO REM LO X A, $[X]$; RESULT LO TO REM LO SBIT O, $[B]$; SET QUOTIENT BIT C616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO					
X A, [B+] IFBNE #4 JP DV616L SC ; RESET BORROW LD A, [X+] ; TEST FIELD LO TO ACC SUBC A, [B] ; SUBT DR LO FROM REM LO LD A, [X] ; TEST FIELD HI TO ACC LD B,#5 SUBC A, [B] ; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JP DV616T ; FROM SUBTRACTION X A, [X-] ; SUBT RESULT HI TO REM HI LD A, [X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A, [B] ; SUBT DR LO FROM REM LO X A, [X] ; RESULT LO TO REM LO X A, [X] ; RESULT LO TO REM LO SBIT O, [B] ; SET QUOTIENT BIT G16T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO	V616L:				
IFBNE#4JPDV616LSC:LDA, [X+]:TEST FIELD LO TO ACCSUBCA, [B]:SUBT DR LO FROM REM LOLDA, [X]:TEST FIELD HI TO ACCLDB,#5SUBCA, [B]:SUBT DR HI FROM REM HIIFNC:XA, [X-]:SUBT ACTIONXA, [X-]:SUBT RESULT HI TO REM HILDA, [X]:TEST FIELD LO TO ACCLDB,#4SUBCA, [B]:SUBT DR LO FROM REM LOXA, [X]:RESULT LO TO REM LOLDB,#0SBITO, [B]:SET QUOTIENT BIT:DCREMENT AND TESTJPDV616S:CNTR FOR ZERO				;	AND TEST FIELD
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$					
$\begin{array}{cccccccccccccccccccccccccccccccccccc$					
LD A, $[X+]$; TEST FIELD LO TO ACC SUBC A, $[B]$; SUBT DR LO FROM REM LO LD A, $[X]$; TEST FIELD HI TO ACC LD B, $\#5$ SUBC A, $[B]$; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JF DV616T ; FROM SUBTRACTION X A, $[X-]$; SUBT RESULT HI TO REM HI LD A, $[X]$; TEST FIELD LO TO ACC LD B, $\#4$ SUBC A, $[B]$; SUBT DR LO FROM REM LO X A, $[X]$; RESULT LO TO REM LO LD B, $\#0$ SBIT O, $[B]$; SET QUOTIENT BIT 'G16T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO			DV616L		
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$					
LD A, $[X]$; TEST FIELD HI TO ACC LD B,#5 SUBC A, $[B]$; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JP DV616T ; FROM SUBTRACTION X A, $[X-]$; SUBT RESULT HI TO REM HI LD A, $[X]$; TEST FIELD LO TO ACC LD B,#4 SUBC A, $[B]$; SUBT DR LO FROM REM LO X A, $[X]$; RESULT LO TO REM LO LD B,#0 SBIT O, $[B]$; SET QUOTIENT BIT C616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO					
LD B,#5 SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JP DV616T ; FROM SUBTRACTION X A,[X-] ; SUBT RESULT HI TO REM HI LD A,[X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT DR LO FROM REM LO X A,[X] ; RESULT LO TO REM LO LD B,#0 SBIT O,[B] ; SET QUOTIENT BIT C616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO					
SUBC A,[B] ; SUBT DR HI FROM REM HI IFNC ; TEST IF BORROW JF DV616T ; FROM SUBTRACTION X A,[X-] ; SUBT RESULT HI TO REM HI LD A,[X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT DR LO FROM REM LO X A,[Z] ; RESULT LO TO REM LO SBIC A,[B] ; SUBT DR LO FROM REM LO X A,[Z] ; RESULT LO TO REM LO SBIT O,[B] ; SET QUOTIENT BIT C616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO				;	TEST FIELD HI TO ACC
IFNC ; TEST IF BORROW JP DV616T ; FROM SUBTRACTION X A,[X-] ; SUBT RESULT HI TO REM HI LD A,[X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT DR LO FROM REM LO X A,[X] ; RESULT LO TO REM LO LD B,#0 SBIT O,[B] ; SET QUOTIENT BIT 616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO					מווסת הס ווד הסמי הוו מת
JP DV616T ; FROM SUBTRACTION X A,[X-] ; SUBT RESULT HI TO REM HI LD A,[X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT DR LO FROM REM LO X A,[X] ; RESULT LO TO REM LO LD B,#0 SBIT O,[B] ; SET QUOTIENT BIT G616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO			А,[Ď]		
X A,[X-] ; SUBT RESULT HI TO REM HI LD A,[X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT DR LO FROM REM LO X A,[X] ; RESULT LO TO REM LO LD B,#0 SBIT O,[B] ; SET QUOTIENT BIT G616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO			DVGICT		
LD A,[X] ; TEST FIELD LO TO ACC LD B,#4 SUBC A,[B] ; SUBT DR LO FROM REM LO X A,[X] ; RESULT LO TO REM LO LD B,#0 SBLT O,[B] ; SET QUOTIENT BIT G616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO					
LD B,#4 SUBC A,[B] ; SUBT DR LO FROM REM LO X A,[X] ; RESULT LO TO REM LO LD B,#0 SBIT O,[B] ; SET QUOTIENT BIT G616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO					
SUBC A,[B] ; SUBT DR LO FROM REM LO X A,[X] ; RESULT LO TO REM LO LD B,#0 SBIT O,[B] ; SET QUOTIENT BIT 616T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO				;	TO IU ACC
X A,[X] ; RESULT LO TO REM LO LD B,#0 SBIT O,[B] ; SET QUOTIENT BIT G16T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO					SUBT DR LO FROM REM LO
LD B,#0 SBIT O,[B] ; SET QUOTIENT BIT G16T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO					
SBIT O,[B] ; SET QUOTIENT BIT G16T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO				,	LEGGER DO TO REM DO
G16T: DRSZ CNTR ; DECREMENT AND TEST JP DV616S ; CNTR FOR ZERO				•	SET QUOTIENT BIT
JP DV616S ; CNTR FOR ZERO	V616T:				
			2.0200		
				,	

		UCTION CYCLES UCTION CYCLES		
	DIVIDEND DIVISOR I	N [5,4]	(D	עס) אר
		IN [1,0] IN [3,2]		QUOT) TEST FIELD)
DX1616:	LD	CNTR,#16	;	LOAD CNTR WITH LENGTH
	LD	B,#5	;	
	LD	A,[B]		REPLACE DIVISOR WITH
	XOR	A,#OFF	;	
	X LD	A,[B_]	;	DIVISOR TO ALLOW
	XOR	A,[B] A,#OFF	;	OPTIONAL ADDITION OF DIVISOR'S COMPLEMENT IN MAIN PROG. LOOP CLEAR
	X	A, [B-]	,	IN MAIN PROC LOOP
	LD	[B-],#0	:	CLEAR
	LD	[B],#0	;	TEST FIELD
DX616S:	LD	[b],#0 В,#0	,	
DX616L:	RC	- , , •		
	LD	A,[B]		
	ADC	A,[B]	;	LEFT SHIFT DIVIDEND LO
	Х	A,[B+]	-	
	LD	A,[B]		
	ADC	A,[B]	;	LEFT SHIFT DIVIDEND HI
	Х	A,[B+]		
	LD	A,[B]		
	ADC	A,[B]	;	LEFT SHIFT TEST FIELD LO
	X	A,[B+]		
	LD	A,[B]		
	ADC	A,[B]	;	LEFT SHIFT TEST FIELD HI
	X	A,[B+]		
	SC LD	A FD1		DIVISORY (DRV) IO TO AGO
	LD	A,[B] B,#2		DIVISORX (DRX) LO TO ACC (l'S COMPLEMENT)
	ADC	A,[B]	;	ADD REM LO TO DRX LO
	LD	B,#5	,	
	LD	A,[B]	:	DIVISORX (DRX) HI TO ACC
	LD	B,#3	;	
	ADC	A,[B]	;	ADD REM HI TO DRX HI
	IFNC		;	TEST IF NO CARRY FROM
	JP	DX616T	;	1'S COMPL.ADDITION
	Х	A,[B+]		RESULT TO REM HI
	LD	A,[B]	;	DRX LO TO ACCUMULATOR
	LD	B,#2		
	ADC	A,[B]		ADD REM LO TO DRX LO
	X	A,[B]	;	RESULT TO REM LO
	LD	B,#0	-	
	SBIT DRSZ	0,[B]		SET QUOTIENT BIT DECREMENT AND TEST
	JP	CNTR DX616L	;	CNTR FOR ZERO
	JP RET	DVOTOT	;	RETURN FROM SUBROUTINE
DX616T:	DRSZ	CNTR	,	DECREMENT AND TEST
	JMP	DX616S	;	CNTR FOR ZERO
	RET		;	RETURN FROM SUBROUTINE

	DIVISOR I QUOTIENT	IN [3,2,1,0] IN [5,4] IN [1,0] R IN [3,2]	(DD) (DR) (QUOT) (TEST FIELD)	
)V2815:)28155:	LD	CNTR,#16 B,#0	; LOAD CNTR WITH LENGTH OF QUOTIENT FIELD	
2815L:	RC LD	A,[B]		
	ADC		; LEFT SHIFT LOWER	
	X		; LEFT SHIFT LOWER ; BYTE OF DIVIDEND	
	LD ADC	A,[B] A,[B]	. IPPT CUIPT NEVT UICUPD	
	X	A,[B+]	; LEFT SHIFT NEXT HIGHER ; BYTE OF DIVIDEND	
	LD	V LBJ		
	ADC	A,[B]	; LEFT SHIFT NEXT HIGHER ; BYTE OF DIVIDEND	
	X LD	A,[B+] A,[B]	; BYTE OF DIVIDEND	
	ADC	A, [B]	; LEFT SHIFT UPPER	
	X	A,[B-]	; BYTE OF DIVIDEND	
			RY WOULD REQUIRE THAT THE DIVISOR BE SUBTRACTED OWN WITH THE DIV168*** SUBROUTINE. ; REM LOWER BYTE TO ACC	
	SC	A ,[D]	; TEST SUBTRACT LOWER	
	LD	B,#4	; BYTE OF DR FROM	
	SUBC LD	A,[B] B,#3	; LOWER BYTE OF REM TEST SUBTRACT UPPER	
	LD	A,[B]	; BYTE OF DIVISOR	
	LD	B,#5	; LOWER BYTE OF REM ; TEST SUBTRACT UPPER ; BYTE OF DIVISOR ; FROM UPPER BYTE ; OF REMAINDER	
			; OF REMAINDER	
	SUBC	м,[D]		
		D2815T	; TEST IF BORROW ; FROM SUBTRACTION	
	SUBC IFNC JP LD	D2815T B,#3	; FROM SUBTRACTION ; UPPER BYTE OF RESULT	
	SUBC IFNC JP LD X	D2815T B,#3 A,[B+]	; TEST IF BURKOW ; FROM SUBTRACTION ; UPPER BYTE OF RESULT ; TO UPPER BYTE OF REM ; DB LOWER BYTE TO ACC	
	SUBC IFNC JP LD	D2815T B,#3 A,[B+] A,[B] B,#2	; TEST IF BURKOW ; FROM SUBTRACTION ; UPPER BYTE OF RESULT ; TO UPPER BYTE OF REM ; DR LOWER BYTE TO ACC ; SUBTRACT LOWER BYTE	
	SUBC IFNC JP LD X LD LD X	D2815T B,#3 A,[B+] A,[B] B,#2	DR LOWER BYTE TO ACC SUBTRACT LOWER BYTE	
	SUBC IFNC JP LD X LD LD X SUBC	D2815T B,#3 A,[B+] A,[B] B,#2	; TEST IF BURKOW ; FROM SUBTRACTION ; UPPER BYTE OF RESULT ; TO UPPER BYTE OF REM ; DR LOWER BYTE TO ACC ; SUBTRACT LOWER BYTE ; OF DIVISOR FROM ; LOWER BYTE OF BEMAINDER	
	SUBC IFNC JP LD X LD LD X	D2815T B,#3 A,[B+] A,[B] B,#2	; TEST IF BORKOW ; FROM SUBTRACTION ; UPPER BYTE OF RESULT ; TO UPPER BYTE OF REM ; DR LOWER BYTE TO ACC ; SUBTRACT LOWER BYTE ; OF DIVISOR FROM ; LOWER BYTE OF ; REMAINDER	
	SUBC IFNC JP LD X LD LD X SUBC X LD SBIT	D2815T B,#3 A,[B+] A,[B] B,#2 A,[B] A,[B] A,[B] B,#0 O,[B]	; LOWER BYTE OF ; REMAINDER ; SET QUOTIENT BIT	
	SUBC IFNC JP LD X LD LD X SUBC X LD SBIT DRSZ	D2815T B,#3 A,[B+] A,[B] B,#2 A,[B] A,[B] A,[B] B,#0 O,[B] CNTR	; LOWER BYTE OF ; REMAINDER ; SET QUOTIENT BIT ; DECREMENT AND TEST	
	SUBC IFNC JP LD X LD LD X SUBC X LD SBIT	D2815T B,#3 A,[B+] A,[B] B,#2 A,[B] A,[B] A,[B] B,#0 O,[B]	; LOWER BYTE OF ; REMAINDER ; SET QUOTIENT BIT	
02815T:	SUBC IFNC JP LD X LD LD X SUBC X LD SBIT DRSZ JMP RET DRSZ	D2815T B,#3 A,[B+] A,[B] B,#2 A,[B] A,[B] A,[B] B,#0 0,[B] CNTR D2815L CNTR	; LOWER BYTE OF ; REMAINDER ; SET QUOTIENT BIT ; DECREMENT AND TEST ; CNTR FOR ZERO ; RETURN FROM SUBROUTINE ; DECREMENT AND TEST	
02815T:	SUBC IFNC JP LD X LD LD X SUBC X LD SBIT DRSZ JMP RET	D2815T B,#3 A,[B+] A,[B] B,#2 A,[B] A,[B] A,[B] B,#0 O,[B] CNTR D2815L	; LOWER BYTE OF ; REMAINDER ; SET QUOTIENT BIT ; DECREMENT AND TEST ; CNTR FOR ZERO ; RETURN FROM SUBROUTINE	

DX3210-F		DIVISION SUBROU	JTINE
	70 BYTES 1510 INSTR	UCTION CYCLES	AVERAGE
		RUCTION CYCLES	
	DIVIDEND 1	[N [3,2,1,0]	(DD)
	DIVISOR IN		(DR)
		IN [3,2,1,0]	(QUOT)
	REMAINDER	IN [5,4]	(TEST FIELD)
DX3216:	LD	CNTR,#32	; LOAD CNTR WITH LENGTH
DAOSIO.	LD	B,#7	; OF DIVIDEND FIELD
	LD	A,[B]	; REPLACE DIVISOR WITH
	XOR	A,#OFF	; 1'S COMPLEMENT OF
	X	A,[B-]	; DIVISOR TO ALLOW
	LD XOR	A,[B] A,#OFF	OPTIONAL ADDITION OF DIVISOR'S COMPLEMENT
	X	A,[B-]	; IN MAIN PROG. LOOP
	LD	[B-],#O	; CLEAR
	LD	[B],#O	; TEST FIELD
DX326S: DX326L:	LD RC	В,#О	
DAJ201:	LD	A,[B]	
	ADC	A,[B]	; LEFT SHIFT DIVIDEND LO
	X	A,[B+]	
	LD ADC	A,[B] A,[B]	: LEFT SHIFT NEXT HIGHER
	X	A,[B] A,[B+]	; DIVIDEND BYTE
	LD	A,[B]	, 51125215 5112
	ADC	A,[B+]	; LEFT SHIFT NEXT HIGHER
	X	A,[B+]	; DIVIDEND BYTE
	LD ADC	A,[B] A,[B]	; LEFT SHIFT DIVIDEND HI
	X	A,[B+]	, DEFI SHIFI DIVIDEND HI
	LD	A,[B]	
	ADC	A,[B]	; LEFT SHIFT TST FIELD LO
	X	A,[B+]	
	LD ADC	A,[B] A,[B]	; LEFT SHIFT TST FIELD HI
	X	A,[B+]	
	IFC		; **TEST IF BIT SHIFTED
	JP	DX326B	; ** OUT OF TEST FIELD
	SC LD	A,[B]	; DVSORX (DRX) LO TO ACC
	LD	B,#4	; (1'S COMPLEMENT)
	ADC	A,[B]	; ADD REM LO TO DRX LO
	LD	B,#7	
	LD LD	A,[B] B #5	; DVSORX (DRX) HI TO ACC ; (l'S COMPLEMENT)
	ADC	B,#5 A,[B]	; ADD REM HI TO DRX HI
	IFNC	,[-]	; TEST IF NO CARRY FROM
	JP	DX326T	; 1'S COMPL. ADDITION
	X LD	A,[B+]	; RESULT TO REM NI ; DRX LO TO ACCUMULATOR
	LD	A,[B] B,#4	, DIA DO TO ACCOMODATON
DX326R:	ADC	A,[B]	; ADD REM LO TO DRX LO
			; ** ADD REM HI TO DRX HI
	Х	A,[B]	; RESULT TO REM LO
LD		B,#0	; ** RESULT TO REM HI
	SBIT	0,[B]	; SET QUOTIENT BIT
	DRSZ	CNTR	; DECREMENT AND TEST
	JMP	DX326L	; CNTR FOR ZERO : RETURN FROM SUBROUTINE
DX326T:	RET DRSZ	CNTR	; RETURN FROM SUBROUTINE : DECREMENT AND TEST
	JMP	DX326S	; CNTR FOR ZERO
	RET		; RETURN FROM SUBROUTINE
DX326B:	LD	A,[B]	; ** REM LO TO ACC
	LD ADC	B,#6 A,[B]	; ** B PTR TO DRX LO ; ** ADD DRX LO TO REM LO
	X	A,[B]	; ** RESULT TO REM LO
	LD	B,#7	**
	LD	A,[B]	; ** DRX HI TO ACC
	LD JP	B,#5 DX36R	; ** B PTR TO REM HI : **
* *			-
Ar Ar		RUCTIONS UNNEC 2**15 (DX3215	ESSARY IF DIVISOR

		RVES AS EXAMPLE	
	05 51 51 1	DIVISION SUBROU	TNE
		40 BYTES	
		MINIMAL C	DDE
		3879 INST	RUCTION CYCLES AVERAGE
		4535 INST	RUCTION CYCLES MAXIMUM
		IN [3,2,1,0]	(DD)
		N [9,8,7] IN [3,2,1,0]	(DR) (QUOT)
		IN [6,5,4]	(TEST FIELD)
0V3224:	LD	CNTR,#32	; LOAD CNTR WITH LENGTH
0000010	LD	B,#6	; OF DIVIDEND FIELD
CLRLUP:	LD	[B-],#0	; CLEAR TEST FIELD
	IFBNE	#3	; TOP OF DIVIDEND FIELD
	JP	CLRLUP	
DVSHFT:	RC		
	LD	B,#0	
SHFTLP:	LD	A,[B]	
	ADC	A,[B]	; LEFT SHIFT DIVIDEND
	Х	A,[B+]	; AND TEST FIELD
	IFBNE	#7	; BOTTOM OF DR FIELD
	JP	SHFTLP	
	IFC		; TEST IF BIT SHIFTED
	JP	DVSUBT	; *** OUT OF TEST FIELD
	SC		; RESET BORROW
	LD	X,#4	
ISTLUP:	LD	A,[X+]	; TEST SUBTRACT DIVISOR
	SUBC	A,[B]	; FROM TEST FIELD
	LD	A,[B+]	; INCREMENT B POINTER
	IFBNE JP	#10 TSTLUP	; TOP OF DIVISOR + 1
	JF IFNC	TOTTOL	; TEST IF BORROW
	JP	DVTEST	; IESI IF BORROW : FROM SUBTRACTION
	LD	B,#7	, FROM DODINACTION
DVSUBT:	LD	X,#4	
SUBTLP:	LD	A,[X]	; SUBTRACT DIVISOR
•	SUBC	A,[B]	; FROM REMAINDER
	X	A,[X+]	; IN TEST FIELD
	LD	A,[B+]	; INCREMENT B POINTER
	IFBNE	#10	; TOP OF DIVISOR + 1
	JP	SUBTLP	
	LD	B,#0	
	SBIT	0,[B]	; SET QUOTIENT BIT
OVTEST:	DRSZ	CNTR	; DECREMENT AND TEST
	JP	DVSHFT	; CNTR FOR ZERO
	RET		; RETURN FROM SUBROUTINE
ST:	SBIT DRSZ JP	O,[B] CNTR	; DECREMENT AND TEST ; CNTR FOR ZERO

4.0 DECIMAL (PACKED BCD)/BINARY CONVERSION

Subroutines For Two Byte Conversion:

- DECBIN - Decimal (Packed BCD) to Binary
 - 24 Bytes ***
 - 1030 Instruction Cycles
- Fast Decimal (Packaged BCD) to Binary FDTOB
 - 76 Bytes - 92 Instruction Cycles
- BINDEC - Binary to Decimal (Packed BCD)
 - 25 Bytes ***
 - 856 Instruction Cycles

- Fast Binary to Decimal (Packed BCD) FBTOD — 59 Bytes
 — 334 Instruction Cycles
- VFBTOD Very Fast Binary to Decimal (Packed BCD) - 189 Bytes
 - 144 Instruction Cycles Average
 - 208 Instruction Cycles Maximum

***These subroutines extendable to multiple byte conversion by simply changing parameters within subroutine as shown, with number of bytes in subroutine remaining constant.

-		epresents very minimal decimal number of any		
ALGORITH	M:			
decimal nu decimal ope bit position, shifting dow field. The re order bit in each nibble tain a high right shifts ti rects the re repeated, w number of ti 16 Bit: Bina Pac 24 Bit: Bina Pac 32 Bit: Bina	mber. During e erand and the b with the low o wn into the high esidual decimal each of its nib in the BCD op order bit equal the BCD operar esult to BCD for ith the total nur- bit positions in the	4, 3]]	rithm, the right one I operand the binary for a high cted from nd to con- effectively then cor- le is ther	a 3 4 7 7 7 7 7
	ction Cycles (1	6 Bit)		
DECBIN:	LD	CNTR,#16	;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;	LOAD CNTR WITH NUMBER OF BIT POSITIONS IN BCD FIELD #16 FOR 16 BIT (2 BYTE) #'S 24/32 FOR 24/32 BIT
DB1:	LD RC	B,#3	;	#'S 5/7 FOR 24/32 BIT
DB2:	LD RRC X IFBNE JP LD SC	A,[B] A A,[B-] #OF DB2 B,#3	;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;	PROGRAM LOOP TO RIGHT SHIFT DECIMAL (BCD) AND BINARY FIELDS. LOOP JUMP BACK #'S 5/7 FOR 24/32 BIT SET CARRY FOR SUBTRACT
DB3:	LD IFBIT SUBC X IFBNE JP DRSZ JP RET	A, [B] 7, [B] A, #030 3, [B] A, #3 A, [B-] #1 DB3 CNTR DB1	;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;	TEST HIGH ORDER BITS OF ECD NIBBLES, AND SUBTRACT A THREE FROM EACH NIBBLE IF HIGH ORDER BIT OF NIBBLE IS A ONE #'S 2/3 FOR 24/32 BIT LOOP BACK FOR MORE BCD BYTES DECREMENT AND TEST IF CNTR EQUAL TO ZERO RETURN FROM SUBROUTINE

FDTOB—FAS BCD Format:	Four Nik *** [1]	bles – W, X, Y, Z, with = 16W + X = 16Y + Z	hW = Hi	Order Nibble
Algorithm:	BCD IN Temp in Binary ir 76 Bytes	n [4, 3]	ow + x) ·	+ (10Y + Z)
FDTOB:	RC LD AND RRC X RRC ADD X LD AND ADC X LD ADC X ADC X ADC LD X CLR LFC LD X LD AND X LD AND X LD ADC X ADC ADC X ADC ADC X ADC ADC ADC X ADC ADC X ADC ADC X ADC ADC X ADC ADC X ADC ADC ADC X ADC ADC X ADC ADC X ADC ADC X ADC ADC X ADC X ADC ADC X ADC ADC X ADC X ADC ADC X ADC ADC X ADC ADC X ADC ADC X ADC ADC ADC ADC ADC ADC ADC ADC ADC ADC	B,#1 A,[B+] A,#OFO A A,[B] A,[B] A,[B] A,[B] A,[B] A,[B] A,[B] A,[B] A,[B] A,[B] A,[B] A,[B] B,#3 A,[B+] A,[B] A,[B-] A,[B] A,[B-] A,[B] A,[B-] A,[B]	* * * * * * * * * * * * * * * * * * * *	8W 8W TO TEMP 4W 2W 2W + 8W = 10W 10W TO TEMP 16W + X EXTRACT X 10W + X 10W + X TO TEMP 2. (10W + X) 2. (10W + X) TO TEMP 3. (10W + X) = 16P + Q 16P + Q TO [3] 16C TO A (C = CARRY) 16C TO [4] 16P + Q 16Q + P 16Q + P 16Q + P 16Q + P 16Q + P 16C + P TO [3] 16Q + P EXTRACT P 16C + P 16C + P TO [4]** 16Q + P EXTRACT 16Q 16Q TO [3]** 2. (10W + X)

	Х	A,[B+]	;	2 BYTE 2.(10W + X)
	CLR	A,[B-]	;	ADD: + 48.**(10W + X)
	ADC	A,[B]	;	16C + P + NU C
	X	A,[B-]	;	50.(10W + X)
	LD	A,[B]		
	ADC	A,[B]	;	DOUBLE
	X	A,[B+]	;	50.(10W + X)
	LD	A,[B]	;	TO FORM
	ADC	A,[B]	;	100.(10W + X)
	X	A,[B]	;	IN [3,4]
	LD	B,#O		
	LD	A,[B]	;	16Y + Z
	AND	A,#OFO	;	EXTRACT 16Y
	LD	B,#2		
	RRC	A	;	84
	Х	A,[B]	;	8Y TO TEMP
	LD	A,[B]		
	RRC	A	;	4Y
	RRC	A	;	24
	ADC	A,[B]	;	2Y + 8Y = 10Y
	Х	A,[B]	;	10Y TO TEMP
	LD	B,#0		
	LD	A,[B]	;	16Y + Z
	AND	A,#OF	;	EXTRACT Z
	LD	B,#2		
	ADD	A,[B]	;	10Y + Z
	LD	B, #3		
	ADC	A,[B]	;	TWO BYTE ADD
	Х	A,[B+]	;	100.(10W + X)
	CLR	A	;	+ (10Y + Z)
	ADC	A,[B]	;	WITH BINARY
	Х	A,[B]	;	RESULT TO [3,4]
	RET			
- 1				

Г

	Binary to Decin	nal (Packed BCD)		
	-	epresents very minimal	code for	
		er of any length to pack		
decimal.	,	, , , , ,		
ALGORITH	M:			
The packed	BCD decimal	result is resident just al	bove the	
		nt number of bytes must		
		ouring each cycle of the a		
the binary n	umber is shifted	left one bit position. The	e packed	I
		shifted left one bit posit		
•		ary field being shifted up		
		e BCD field. The shifted		
		corrected by using the D lition an "ADD A, #066"		
		unction with the DCOR		
		ntire cycle is then repea	•	
the total nu	mber of cycles	being equal to the numb	per of bit	t
positions in	the binary field	l.		
16 Bit:	Binary	in [1, 0]		
	Packed	d BCD in [4, 3, 2]		
24 Bit:	Binary	in [2, 1, 0]		
	Packed	d BCD in [6, 5, 4, 3]		
32 Bit:	Binary	in [3, 2, 1, 0]		
	Packed	BCD in [8, 7, 6, 5, 4]		
25 Bytes				
856 Instruct	tions Cycles (16	6 Bit)		
BINDEC:	LD	CNTR,#16	;	LOAD CNTR WITH NUMBER OF BIT POSITIONS
	-		;	IN BINARY FIELD
			;	#16 FOR 16 BIT (2 BYTE)
			;	#'S 24/32 FOR 24/32 BIT
	RC	P //O		
BD1:	LD LD	B,#2	;	#'S 3/4 FOR 24/32 BIT CLEAR BCD FIELD
DD1.	IFBNE	[B+],#0 #5	;	#'S 7/9 FOR 24/32 BIT
	JP	BD1	;	JUMP BACK FOR CLR LOOP
BD2:	LD	B,#0	,	
BD3:	LD	A,[B]	;	PROGRAM LOOP TO
	ADC	A,[B]	;	LEFT SHIFT
	X	A,[B+]	;	BINARY FIELD
	IFBNE JP	#2 BD3	;	#'S 3/4 FOR 24/32 BIT
BD4:	J P LD	виз А,[B]	;	JUMP BACK FOR SHIFT LOOP1 PROGRAM LOOP TO
	ADD	A,#066	;	LEFT SHIFT AND
	ADC	A,[B]	;	DECIMAL CORRECT
	DCOR	A	;	RESULT OF SHIFT
	X	A,[B+]	;	IN BCD FIELD
	IFBNE	#5 8D4	;	#'S 7/9 FOR 24/32 BIT
	JP DRSZ	BD4 CNTR	;	JUMP BACK FOR SHIFT LOOP2 DECREMENT AND TEST IF
	JP	BD2	;	CNTR EQUAL TO ZERO
	RET		;	RETURN FROM SUBROUTINE

-

Algorithm	algorith	gorithm is based on the nm, except that it is opti of execution.	
Binary in Packed E	[1, 0] 3CD in [4, 3, 2]		
59 Bytes			
334 Instr	uction Cycles		
FBTOD:	RC LD	B,#1	
	LD	A,[B]	
	SWAP	А,[b] А	: REVERSE NIBBLES IN
	X	A,[B]	; REVERSE NIBBLES IN ; UPPER BINARY BYTE
	LD	A,[B+]	; EXTRACT ORIGINAL UPPER
	AND	A,#OF	; NIBBLE OF HI BYTE
	IFGT	A,#9	; IF NIBBLE GREATER THAN
	ADD	A,#06	; NINE, THEN ADD SIX TO CORRECT BCD NIBBLE
	X	A,[B+]	; NIBBLE TO LOWER BCD BYTE
	LD	[B+],#O	; CLEAR UPPER BCD BYTES
	LD	[B],#0	; INITIALIZE CNTR TO COVER
	LD	CNTR,#4	; REMAINING HI NIBBLE (ORIGINALLY LO NIBBLE)
		0	; IN UPPER BINARY BYTE
FBD1:	LD	B,#1	; PROGRAM LOOP TO
	LD	A,[B]	; LEFT SHIFT A BIT
	ADC	A,[B]	OUT OF UPPER BINARY
	X	A,[B+]	; BYTE INTO LOW ORDER
	LD	A,[B]	; BIT POSITION OF BCD
	ADD	A,#066	; FIELD, AS LOWER TWO
	ADC	A,[B]	; BYTES OF BCD FIELD
	DCOR	A	ARE LEFT SHIFTED WITH
	Х	A,[B+]	THE LOWER BYTE BEING
	LD	A,[B]	; DECIMAL CORRECTED
	ADC	A,[B]	; MIDDLE BYTE OF BCD FIELD
	Х	A,[B]	, NEED NOT BE DECIMAL CORRECTED, SINCE
			; MAX VALUE IS 2 (256)
	DRSZ	CNTR	; DECREMENT AND TEST IF
	JP	FBD1	; CNTR EQUAL TO ZERO
	LD	CNTR,#8	; INITIALIZE CNTR TO COVER
FBD2:	LD	B,#0	; LOWER BINARY BYTE
	LD	A,[B]	PROGRAM LOOP TO
	ADC	A,[B]	LEFT SHIFT A BIT
	Х	A,[B]	; OUT OF LOWER BINARY
	LD	B,#2	; BYTE INTO LOW ORDER
	LD	A,[B]	; BIT POSITION OF BCD
	ADD	A,#066	; FIELD, AS BCD FIELD
	ADC	A,[B]	; IS LEFT SHIFTED WITH
	DCOR	A	; THE LOWER TWO BYTES
	Х	A,[B+]	; OF THE FIELD BEING
	LD	A,[B]	; DECIMAL CORRECTED
	ADD	A,#066	; ADD (NOT ADC) HEX 66
	ADC	A,[B]	; TO SET UP "ADD" DCOR
	DCOR	A	; DECIMAL CORRECT MIDDLE
	Х	A,[B+]	; BYTE OF BCD FIELD
	LD	A,[B]	; UPPER BYTE OF BCD FIELD
	ADC	A,[B]	; NEED NOT BE DECIMAL
	Х	A,[B]	; CORRECTED, SINCE MAX
			; VALUE IS 6 (65535)
	DRSZ	CNTR	; DECREMENT AND TEST IF
	JP	FBD2	; CNTR EQUAL TO ZERO
	RET		; RETURN FROM SUBROUTINE

		tion in BCD of power)	
	one's o der to	at binary field (2 bytes complemented by prog facilitate bypass brand d bit in the binary fie o zero	gram, in or ching wher	-	
	in [1, 0] [4, 3, 2]	2010.			
	tes struction Cycles struction Cycles	•			
VFBTOD:	RC				
	LD	B,#0			
	LD AND	A,[B] A,#OF	•	EXTRACT LO NIBBLE	
	IFGT	A,#9	;	TEST NIBBLE 9	
	ADD	A,#6	;	ADD 6 FOR CORRECTION	
	LD	B,#2			
	X	A,[B+]	;	STORE IN LO BCD NIBBLE	
	LD	[B+],#0	;	CLEAR UPPER BCD NIBBLES	
	LD LD	[B],#O B,#l	;	RCD NIRRTE2	
	LD	A,[B]			
	XOR	A,#OFF	;	COMPLEMENT HI BYTE	
	Х	A,[B-]	;	FOR REVERSE TESTING	
	LD	A,[B]	;	OF BINARY NUMBER	
	XOR X	A,#OFF	;	COMPLEMENT LO BYTE FOR REVERSE TESTING	
	IFBIT	A,[B] 4,[B]	;	TEST BINARY BIT 4	
	JP	VFB1	;	TO CONDITIONALLY	
	LD	B,#2	;	ADD BCD 16	
	LD	A,#07C	;	16 + 66	
	ADC	A,[B]	;	ADD BCD 16	
	DCOR X	A A,[B]			
	LD	B,#0			
VFB1:	IFBIT	5,[B]	;	TEST BINARY BIT 5	
	JP	VFB2	;	TO CONDITIONALLY	
	LD	B,#2	;	ADD BCD 32	
	LD	A,#098	;	32 + 66	
	ADC DCOR	A,[B] A	;	ADD BCD 32	
	X	А А,[B]			
	LD	B,#0			
VFB2:	IFBIT	6,[B]	;	TEST BINARY BIT 6	
	JP	VFB3	;	TO CONDITIONALLY	
	LD	B,#2	;	ADD BCD 64	
	LD ADC	A,#OCA A,[B]	;	64 + 66 ADD BCD 64	
	DCOR	A,[b]	,		
	x	A,[B+]			
	CLR	A			
	ADC	A,[B]	;	ADD CARRY	
	X LD	A,[B] B #0			
	עם	В,#О			

VFB3:	IFBIT	7,[B]	; TEST BINARY BIT 7
	JP	VFB4	; TO CONDITIONALLY
	LD	B,#2	; ADD BCD 128
	LD	A,#08E	; 28 + 66
	ADC	A,[B]	; ADD BCD 28
			, MDD DOD 20
	DCOR	A	
	Х	A,[B+]	
	LD	A,#1	
	ADC	A,[B]	; ADD BCD 1
	X	A,[B]	,
VFB4:	LD	B,#1	; HI BINARY BYTE
	IFBIT	0,[B]	; TEST BINARY BIT 8
	JP	VFB5	; TO CONDITIONALLY
	LD	B,#2	ADD BCD 256
	LD		•
		A,#OBC	; 56 + 66
	ADC	A,[B]	; ADD BCD 56
	DCOR	A	
	Х	A,[B+]	
	LD	A,#2	
	ADC	A,[B]	; ADD BCD 2
	Х	A,[B]	
	LD	B,#1	
UPDE -			MECH DINADY DIM O
VFB5:	IFBIT	1,[B]	; TEST BINARY BIT 9
	JP	VFB6	; TO CONDITIONALLY
	LD	B,#2	; ADD BCD 512
	LD	A,#078	; $12 + 66$
	ADC		
		A,[B]	; ADD BCD 12
	DCOR	A	
	Х	A,[B+]	
	LD	A,#06B	; 5 + 66
	ADC	A,[B]	; ADD BCD 5
	DCOR	A	
	Х	A,[B]	
	LD	B,#1	
VFB6:	IFBIT	2,[B]	; TEST BINARY BIT 10
	JP	VFB7	; TO CONDITIONALLY
	LD	B,#2	ADD BCD 1024
	LD	A,#08A	
	ADC	A,[B]	; ADD BCD 24
	DCOR	A	
	X	A,[B+]	
	LD	A,#076	; 10 + 66
	ADC	A,[B]	; ADD BCD 10
	DCOR	A	
	Х	A,[B]	
	LD	B,#1	
	IFBIT	3,[B]	; TEST BINARY BIT 11
VF87•	TLDTT		-
VFB7:	TD		
VFB7:	JP	VFB8	; TO CONDITIONALLY
VFB7:	JP LD	VFB8 B,#2	; TO CONDITIONALLY ; ADD BCD 2048
VFB7:		B,#2	; ADD BCD 2048
VFB7:	LD LD	B,#2 A,#OAE	; ADD BCD 2048 ; 48 + 66
VFB7:	LD LD ADC	B,#2 A,#0AE A,[B]	; ADD BCD 2048
VFB7:	LD LD ADC DCOR	B,#2 A,#0AE A,[B] A	; ADD BCD 2048 ; 48 + 66
VFB7:	LD LD ADC	B,#2 A,#OAE A,[B] A A,[B+]	; ADD BCD 2048 ; 48 + 66
VFB7:	LD LD ADC DCOR	B,#2 A,#OAE A,[B] A A,[B+]	; ADD BCD 2048 ; 48 + 66
VFB7:	LD LD ADC DCOR X LD	B,#2 A,#0AE A,[B] A A,[B+] A,#086	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VFB7:	LD LD ADC DCOR X LD ADC	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B]	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48
VF87:	LD LD ADC DCOR X LD ADC DCOR	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VF87:	LD LD ADC DCOR X LD ADC DCOR X	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A A,[B]	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VF87:	LD LD ADC DCOR X LD ADC DCOR	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VF87:	LD LD ADC DCOR X LD ADC DCOR X	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A A,[B]	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VFB7:	LD LD ADC DCOR X LD ADC DCOR X	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A A,[B]	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VFB7:	LD LD ADC DCOR X LD ADC DCOR X	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A A,[B]	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VFB7:	LD LD ADC DCOR X LD ADC DCOR X	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A A,[B]	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VFB7:	LD LD ADC DCOR X LD ADC DCOR X	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A A,[B]	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VF87:	LD LD ADC DCOR X LD ADC DCOR X	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A A,[B]	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VFB7:	LD LD ADC DCOR X LD ADC DCOR X	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A A,[B]	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VFB7:	LD LD ADC DCOR X LD ADC DCOR X	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A A,[B]	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VFB7:	LD LD ADC DCOR X LD ADC DCOR X	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A A,[B]	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66
VF87:	LD LD ADC DCOR X LD ADC DCOR X	B,#2 A,#0AE A,[B] A A,[B+] A,#086 A,[B] A A,[B]	; ADD BCD 2048 ; 48 + 66 ; ADD BCD 48 ; 20 + 66

VFB8:	IFBIT	4,[B]	; TEST BINARY BIT 12	
	JP	VFB9	; TO CONDITIONALLY	
	LD	B,#2	; ADD BCD 4096	
	LD	A,#OFC	; 96 + 66	
	ADC	A,[B]	; ADD BCD 96	
	DCOR	A		
	X	A,[B+]		
	LD	A,#0A6	; 40 + 66	
	ADC	A,[B]	; ADD BCD 40	
	DCOR	A		
	X	A,[B]		
VFB9:	LD IFBIT	B,#1 5 [B]	; TEST BINARY BIT 13	
VED9:	JP	5,[B] VFB10		
	LD	B,#2	; TO CONDITIONALLY ; ADD BCD 8192	
	LD	A,#0F8	; $92 + 66$	
	ADC	A,[B]	; ADD BCD 92	
	DCOR	A		
	X	A,[B+]		
	LD	A,#0E7	; 81 + 66	
	ADC	A,[B]	; ADD BCD 81	
	DCOR	A		
	X	A,[B]		
	CLR	A		
	ADC	A,[B]	; ADD CARRY	
	X	A,[B]		
	LD	B,#1		
VFB10:	IFBIT	6,[B]	; TEST BINARY BIT 14	
	JP	VFB11	; TO CONDITIONALLY	
	LD	B,#2	; ADD BCD 16384	
		A,#OEA	; 84 + 66	
	ADC	A,[B]	; ADD BCD 84	
	DCOR X	A A,[B+]		
	r TD	A,[D+] A,#0C9	; 63 + 66	
	ADC	A,#009 A,[B]	; ADD BCD 63	
	DCOR	A,[D]	,	
	X	A,[B+]		
	LD	A,#1		
	ADC	A,[B]	; ADD BCD 1	
	Х	A,[B]		
	LD	B,#1		
VFB11:	IFBIT	7,[B]	; TEST BINARY BIT 15	
	RET		; TO CONDITIONALLY	
	LD	B,#2	; ADD BCD 32768	
	LD	A,#OCE	; 68 + 66	
	ADC	A,[B]	; ADD BCD 68	
	DCOR	A A FRIJ		
	X LD	A,[B+] A,#08D	. 27 + 66	
	ADC	A,#08D A,[B]	; 27 + 66 ; ADD BCD 27	
	DCOR	А,[b] А	, א עטע עעא	
	X	A,[B+]		
	LD	A,#3		
	ADC	A,[B]	; ADD BCD 3	
	Х	A,[B]		
	RET			
-				

COP800 MathPak

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

- Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
- A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.



National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.