	***************	
* ESD Lab-1 Part-1 Assembly Lar	nguage Programming	
* Device - 8051		
/****************	***************************************	
* Student Name: Rushi James	Macwan	
	ystem Design (Spring-19)	
/*****************	*************	
*		
*This code has been written and prod	uced by Rushi James Macwan with the help of	
*Keil Software and Intel 8051 instruction set keeping in mind the CU Honor Code.		
*Credits and courtesy to all the third-	party software platforms.	
*		
*Credits and courtesy to the TA (Trista	an Lennertz) for suggesting me some	
*improvements to my code during the lab sign-off and actually helping me to		
*make it even better. I sincerely thank	k him for his valuable support and	
*guidance.	··	
*		
/*****************	*************	
*Copyright (C) 2019 by Rushi James M		
*		
*Redistribution, modification or use o	f this software in source or binary	
*forms is permitted as long as the files maintain this copyright. Users are		
*permitted to modify this and use it to learn about the field of embedded		
*software. Rushi James Macwan is not liable for any misuse of this material		
*any misuse of this material.	t habite for any misuse of this material	
*		
**********	***************************************	
/*************************************	************	
•		
*Please Note:		
*Please Note:		
*Please Note:  * *Some of these improvements that we	ere made during the sign-off may not be	
*Please Note:  *  *Some of these improvements that we actually reflected in the initial lab sul	ere made during the sign-off may not be bmission on the D2L platform as well	
*Please Note:  *  *Some of these improvements that we actually reflected in the initial lab sull to the sign-off sheet. I have tried to it.	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible	
*Please Note:  *  *Some of these improvements that we  *actually reflected in the initial lab sul  *on the sign-off sheet. I have tried to  *to the code while finally submitting in  **To the code w	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and	
*Please Note:  *  *Some of these improvements that we  *actually reflected in the initial lab sul  *on the sign-off sheet. I have tried to is  *to the code while finally submitting is  *after the sign-off has taken place. I he	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and umbly look forward to receive appreciation	
*Please Note:  *  *Some of these improvements that we *actually reflected in the initial lab sul *on the sign-off sheet. I have tried to it to the code while finally submitting it *after the sign-off has taken place. I he *for my work and efforts while I truly	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and	
*Please Note:  *  *Some of these improvements that we actually reflected in the initial lab sul on the sign-off sheet. I have tried to to the code while finally submitting in after the sign-off has taken place. I he for my work and efforts while I truly on the sign-off has taken place.	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and umbly look forward to receive appreciation	
*Please Note:  *  *Some of these improvements that we actually reflected in the initial lab sulting in the sign-off sheet. I have tried to in the code while finally submitting in the sign-off has taken place. I he for my work and efforts while I truly thank you.	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and umbly look forward to receive appreciation admit that I have been running a bit late.	
*Please Note:  *  *Some of these improvements that we actually reflected in the initial lab sulting in the sign-off sheet. I have tried to in the code while finally submitting in the sign-off has taken place. I he for my work and efforts while I truly thank you.	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and umbly look forward to receive appreciation	
*Please Note:  *  *Some of these improvements that we actually reflected in the initial lab sulting in the sign-off sheet. I have tried to in the code while finally submitting in the sign-off has taken place. I he for my work and efforts while I truly thank you.	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and umbly look forward to receive appreciation admit that I have been running a bit late.	
*Please Note:  *  *Some of these improvements that we  *actually reflected in the initial lab sul  *on the sign-off sheet. I have tried to  *to the code while finally submitting ir  *after the sign-off has taken place. I h  *for my work and efforts while I truly  *Thank you.  *  ********************************	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and umbly look forward to receive appreciation admit that I have been running a bit late.	
*Please Note:  *  *Some of these improvements that we actually reflected in the initial lab sulting in the sign-off sheet. I have tried to in the code while finally submitting in the sign-off has taken place. I he for my work and efforts while I truly thank you.	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and umbly look forward to receive appreciation admit that I have been running a bit late.	
*Please Note:  *  *Some of these improvements that we  *actually reflected in the initial lab sul  *on the sign-off sheet. I have tried to  *to the code while finally submitting ir  *after the sign-off has taken place. I h  *for my work and efforts while I truly  *Thank you.  *  ********************************	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and umbly look forward to receive appreciation admit that I have been running a bit late.	
*Please Note:  *  *Some of these improvements that we actually reflected in the initial lab sulting in the sign-off sheet. I have tried to in the code while finally submitting in the sign-off has taken place. I he for my work and efforts while I truly thank you.  *  ********************************	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and umbly look forward to receive appreciation admit that I have been running a bit late.	
*Please Note:  *  *Some of these improvements that we  *actually reflected in the initial lab sul  *on the sign-off sheet. I have tried to  *to the code while finally submitting ir  *after the sign-off has taken place. I h  *for my work and efforts while I truly  *Thank you.  *  ********************************	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and umbly look forward to receive appreciation admit that I have been running a bit late.	
*Please Note:  *  *Some of these improvements that we  *actually reflected in the initial lab sul  *on the sign-off sheet. I have tried to is  *to the code while finally submitting is  *after the sign-off has taken place. I he  *for my work and efforts while I truly  *Thank you.  *  ********************************	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible tonline before the cut-off deadline and umbly look forward to receive appreciation admit that I have been running a bit late.	
*Please Note:  *  *Some of these improvements that we actually reflected in the initial lab sulting in the sign-off sheet. I have tried to in the code while finally submitting in the sign-off has taken place. I he for my work and efforts while I truly thank you.  *  ********************************	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible t online before the cut-off deadline and umbly look forward to receive appreciation admit that I have been running a bit late.	
*Please Note:  *  *Some of these improvements that we  *actually reflected in the initial lab sul  *on the sign-off sheet. I have tried to is  *to the code while finally submitting is  *after the sign-off has taken place. I he  *for my work and efforts while I truly  *Thank you.  *  ********************************	ere made during the sign-off may not be bmission on the D2L platform as well include as many improvements as possible tonline before the cut-off deadline and umbly look forward to receive appreciation admit that I have been running a bit late.	

MOV A, B	// Moving the value of Y into the accumulator
JZ ERROR2	// If the accumulator is zero, then a jump is performed // specifying that the error for a zero divisor ha
occurred	// specifying that the error for a zero divisor ha
MOV A, #20h	/* Enter the value of X in HEX */
/********MULTIPLICA	TION PART BEGINS*******/
//MUL AB - Ideal instruct	ion that could be used otherwise
RLC A multiplication by 4	// Rotating the accumulator contents left twice for
JC Error1 the carry is set	// Assuming that (X*4) exceeds FFh or 255 value in dec
performed specifying	// If the carry is set, the jump to ERROR1 label
. , ,	// that the overflow event has occurred
CLR C otherwise a carry might n	//After the first rotation left, the carry is cleared not be detected after the second rotation
an error	//and it might stop the system from generating
RLC A	// because we need Z = (X*4) / Y
JC Error1 the carry is set	// Assuming that (X*4) exceeds FFh or 255 value in dec
performed specifying	// If the carry is set, the jump to ERROR1 label
, , , , , ,	// that the overflow event has occurred
BACK: MOV 20h, A of 20h in RAM	// Storing the LSB 8-bit value of (X*4) into the memory location
/*********MULTIPLICA	TION PART ENDS*******/
/*******DIVISION PAR	RT BEGINS*******/
//DIV AB - Ideal instruction	on that could be used otherwise
MOV R0, #0h as a counter	// R0 register is stored with a value of 0h and will work

If the remainder is zero)  // extra jump not required - let it just jump once and calculate the division accordingly  JC STAGE2 // Check for the value of Accumulator carry to be set (in case if the subtraction has resulted into a neg integer) INC R0 // Continuously incrementing the counter for calculating the quotient  JMP HERE  STAGE1: MOV A, R0 // Storing the counter value in the accumulator MOV 21h, A // Storing the value of quotient at 0x21h MOV 22h, #0h // Storing the value of remainder (which is zero) at 0x22h JMP ENDLOOP  STAGE2: ADD A, B // Adding MOV 22h, A // Storing the value of remainder at 0x22h MOV A, R0 // Storing the value of remainder at 0x22h MOV A, R0 // Storing the value of quotient at 0x21h JMP ENDLOOP  /*********DIVISION PART ENDS*********/  ENDLOOP: JMP ENDLOOP  /*********Specifying the error conditions******/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes CLR C // Carry flag is cleared for safety reasons JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error JMP ENDLOOP					
JC STAGE2 // Check for the value of Accumulator carry to be set (in case if the subtraction has resulted into a neg integer) INC R0 // Continuously incrementing the counter for calculating the quotient  JMP HERE  STAGE1: MOV A, R0 // Storing the counter value in the accumulator MOV 21h, A // Storing the value of quotient at 0x21h MOV 22h, #0h // Storing the value of remainder (which is zero) at 0x22h JMP ENDLOOP  STAGE2: ADD A, B // Adding MOV 22h, A // Storing the value of remainder at 0x22h MOV A, R0 // Storing the value of remainder at 0x22h MOV 21h, A // Storing the counter value in the accumulator MOV 21h, A // Storing the value of quotient at 0x21h JMP ENDLOOP  /********DIVISION PART ENDS*********/  ENDLOOP: JMP WHILE // Infinite loop where the MCU keeps jumping between the states  WHILE: JMP ENDLOOP  /********Specifying the error conditions*******/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes CLR C // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error  JMP ENDLOOP	if the remaind		// Check for the value of Accumulator to be zero (in case		
JC STAGE2 // Check for the value of Accumulator carry to be set (in case if the subtraction has resulted into a neg integer) INC R0 // Continuously incrementing the counter for calculating the quotient  JMP HERE  STAGE1: MOV A, R0 // Storing the counter value in the accumulator MOV 21h, A // Storing the value of quotient at 0x21h MOV 22h, #0h // Storing the value of remainder (which is zero) at 0x22h JMP ENDLOOP  STAGE2: ADD A, B // Adding MOV 22h, A // Storing the value of remainder at 0x22h MOV A, R0 // Storing the value of remainder at 0x22h MOV 21h, A // Storing the counter value in the accumulator MOV 21h, A // Storing the value of quotient at 0x21h JMP ENDLOOP  /********DIVISION PART ENDS*********/  ENDLOOP: JMP WHILE // Infinite loop where the MCU keeps jumping between the states WHILE: JMP ENDLOOP  /********Specifying the error conditions******/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes CLR C // Carry flag is cleared for safety reasons JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error JMP ENDLOOP			// extra jump not required - let it just jump once		
JC STAGE2 // Check for the value of Accumulator carry to be set (in case if the subtraction has resulted into a neg integer) INC RO // Continuously incrementing the counter for calculating the quotient  JMP HERE  STAGE1: MOV A, RO // Storing the counter value in the accumulator MOV 21h, A // Storing the value of quotient at 0x21h MOV 22h, #0h // Storing the value of remainder (which is zero) at 0x22h JMP ENDLOOP  STAGE2: ADD A, B // Adding MOV 22h, A // Storing the value of remainder at 0x22h MOV A, RO // Storing the value of remainder at 0x22h MOV 21h, A // Storing the value of quotient at 0x21h JMP ENDLOOP  /*********DIVISION PART ENDS********/  ENDLOOP: JMP WHILE // Infinite loop where the MCU keeps jumping between the states WHILE: JMP ENDLOOP  /********Specifying the error conditions******/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes CLR C // Carry flag is cleared for safety reasons JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error JMP ENDLOOP	and calculate t	the division acco			
case if the subtraction has resulted into a neg integer)  INC R0			- 07		
case if the subtraction has resulted into a neg integer)  INC R0		JC STAGE2	// Check for the value of Accumulator carry to be set (in		
INC RO	case if the sub	traction has resu			
the quotient  JMP HERE  STAGE1: MOV A, R0			<u> </u>		
STAGE1: MOV A, R0	the auotient		,,, ,, ,, ,, ,, ,, ,, ,, ,, ,, ,, ,, ,, ,, ,, ,, ,, ,,, ,,, ,,, ,,, ,,, ,,, ,,		
STAGE1: MOV A, R0	111111111111111111111111111111111111111	JMP HERE			
MOV 21h, A // Storing the value of quotient at 0x21h MOV 22h, #0h // Storing the value of remainder (which is zero) at 0x22h JMP ENDLOOP  STAGE2: ADD A, B // Adding MOV 22h, A // Storing the value of remainder at 0x22h MOV A, RO // Storing the counter value in the accumulator MOV 21h, A // Storing the value of quotient at 0x21h JMP ENDLOOP  /*********DIVISION PART ENDS********  ENDLOOP: JMP WHILE // Infinite loop where the MCU keeps jumping between the states WHILE: JMP ENDLOOP  /*******Specifying the error conditions******/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes CLR C // Carry flag is cleared for safety reasons JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error JMP ENDLOOP		• • • • • • • • • • • • • • • • • • • •			
MOV 21h, A // Storing the value of quotient at 0x21h MOV 22h, #0h // Storing the value of remainder (which is zero) at 0x22h JMP ENDLOOP  STAGE2: ADD A, B // Adding MOV 22h, A // Storing the value of remainder at 0x22h MOV A, RO // Storing the counter value in the accumulator MOV 21h, A // Storing the value of quotient at 0x21h JMP ENDLOOP  /*********DIVISION PART ENDS********  ENDLOOP: JMP WHILE // Infinite loop where the MCU keeps jumping between the states WHILE: JMP ENDLOOP  /*******Specifying the error conditions******/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes CLR C // Carry flag is cleared for safety reasons JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error JMP ENDLOOP	STAGE1:	MOV A. RO	// Storing the counter value in the accumulator		
MOV 22h, #0h // Storing the value of remainder (which is zero) at 0x22h JMP ENDLOOP  STAGE2: ADD A, B // Adding MOV 22h, A // Storing the value of remainder at 0x22h MOV A, RO // Storing the counter value in the accumulator MOV 21h, A // Storing the value of quotient at 0x21h JMP ENDLOOP  /********DIVISION PART ENDS********/  ENDLOOP: JMP WHILE // Infinite loop where the MCU keeps jumping between the states WHILE: JMP ENDLOOP  /*******Specifying the error conditions*****/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes CLR C // Carry flag is cleared for safety reasons JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error JMP ENDLOOP					
STAGE2: ADD A, B  // Adding MOV 22h, A  // Storing the value of remainder at 0x22h MOV A, RO  // Storing the counter value in the accumulator MOV 21h, A  // Storing the value of quotient at 0x21h JMP ENDLOOP  /********DIVISION PART ENDS********  ENDLOOP: JMP WHILE  // Infinite loop where the MCU keeps jumping between the states WHILE: JMP ENDLOOP  /*******Specifying the error conditions******/  ERROR1: MOV 30h, #02h  // Error corresponding to the overflow of bytes CLR C  // Carry flag is cleared for safety reasons JMP ENDLOOP  ERROR2: MOV 30h, #01h  // Zero Divisor (Y = 0) error JMP ENDLOOP			•		
STAGE2: ADD A, B  // Adding     MOV 22h, A			// otoming the range of tennames (times to zero) at onzan		
MOV 22h, A // Storing the value of remainder at 0x22h MOV A, R0 // Storing the counter value in the accumulator MOV 21h, A // Storing the value of quotient at 0x21h JMP ENDLOOP  /*********DIVISION PART ENDS********/  ENDLOOP: JMP WHILE // Infinite loop where the MCU keeps jumping between the states WHILE: JMP ENDLOOP  /*******Specifying the error conditions******/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes CLR C // Carry flag is cleared for safety reasons JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error JMP ENDLOOP		2.12.230.			
MOV 22h, A // Storing the value of remainder at 0x22h MOV A, R0 // Storing the counter value in the accumulator MOV 21h, A // Storing the value of quotient at 0x21h JMP ENDLOOP  /*********DIVISION PART ENDS********/  ENDLOOP: JMP WHILE // Infinite loop where the MCU keeps jumping between the states WHILE: JMP ENDLOOP  /*******Specifying the error conditions******/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes CLR C // Carry flag is cleared for safety reasons JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error JMP ENDLOOP	STAGE2:	ADD A. B	// Adding		
MOV A, R0	0		<del>-</del>		
MOV 21h, A // Storing the value of quotient at 0x21h  JMP ENDLOOP  /********DIVISION PART ENDS********/  ENDLOOP: JMP WHILE // Infinite loop where the MCU keeps jumping between the states  WHILE: JMP ENDLOOP  /******Specifying the error conditions******/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes  CLR C // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error  JMP ENDLOOP					
JMP ENDLOOP  /*******DIVISION PART ENDS********/  ENDLOOP: JMP WHILE // Infinite loop where the MCU keeps jumping between the states  WHILE: JMP ENDLOOP  /******Specifying the error conditions*****/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes  CLR C // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error  JMP ENDLOOP		· · · · · · · · · · · · · · · · · · ·	• • • • • • • • • • • • • • • • • • • •		
/********DIVISION PART ENDS********/  ENDLOOP: JMP WHILE  // Infinite loop where the MCU keeps jumping between the states  WHILE: JMP ENDLOOP  /******Specifying the error conditions*****/  ERROR1: MOV 30h, #02h  // Error corresponding to the overflow of bytes  CLR C  // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h  // Zero Divisor (Y = 0) error  JMP ENDLOOP		•	// otoming the value of quotient at on=2.11		
ENDLOOP: JMP WHILE // Infinite loop where the MCU keeps jumping between the states  WHILE: JMP ENDLOOP  /*******Specifying the error conditions*****/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes  CLR C // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error  JMP ENDLOOP		31111 21122001			
ENDLOOP: JMP WHILE // Infinite loop where the MCU keeps jumping between the states  WHILE: JMP ENDLOOP  /*******Specifying the error conditions*****/  ERROR1: MOV 30h, #02h // Error corresponding to the overflow of bytes  CLR C // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error  JMP ENDLOOP	/********DIVICIONI DADT ENIDC********/				
the states  WHILE: JMP ENDLOOP  /******Specifying the error conditions******/  ERROR1: MOV 30h, #02h  // Error corresponding to the overflow of bytes  CLR C  // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h  // Zero Divisor (Y = 0) error  JMP ENDLOOP	, 5.	VISIOIT FAIRT EIGE	<del>, , , , , , , , , , , , , , , , , , , </del>		
the states  WHILE: JMP ENDLOOP  /******Specifying the error conditions******/  ERROR1: MOV 30h, #02h  // Error corresponding to the overflow of bytes  CLR C  // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h  // Zero Divisor (Y = 0) error  JMP ENDLOOP					
the states  WHILE: JMP ENDLOOP  /******Specifying the error conditions******/  ERROR1: MOV 30h, #02h  // Error corresponding to the overflow of bytes  CLR C  // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h  // Zero Divisor (Y = 0) error  JMP ENDLOOP	ENDLOOP:	IMP WHIIF	// Infinite loop where the MCII keeps jumping between		
WHILE: JMP ENDLOOP  /******Specifying the error conditions******/  ERROR1: MOV 30h, #02h  // Error corresponding to the overflow of bytes  CLR C  // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h  // Zero Divisor (Y = 0) error  JMP ENDLOOP		JIVII VVIIIEE	// illillite loop where the Meo keeps jumping between		
/*******Specifying the error conditions******/  ERROR1: MOV 30h, #02h  // Error corresponding to the overflow of bytes  CLR C  // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h  // Zero Divisor (Y = 0) error  JMP ENDLOOP		IMP FNDI OOP			
ERROR1: MOV 30h, #02h  // Error corresponding to the overflow of bytes  CLR C  // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h  // Zero Divisor (Y = 0) error  JMP ENDLOOP	VVIIIEE.	JIVII LIVDEOOI			
ERROR1: MOV 30h, #02h  // Error corresponding to the overflow of bytes  CLR C  // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h  // Zero Divisor (Y = 0) error  JMP ENDLOOP	/********Opecifying the error conditions******/				
CLR C // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error  JMP ENDLOOP	, spe	and the circle			
CLR C // Carry flag is cleared for safety reasons  JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error  JMP ENDLOOP	FRROR1: MOV	30h #02h	// Frror corresponding to the overflow of hytes		
JMP ENDLOOP  ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error  JMP ENDLOOP					
ERROR2: MOV 30h, #01h // Zero Divisor (Y = 0) error  JMP ENDLOOP			11 carry mag to electrica for surery reasons		
JMP ENDLOOP		J.VII LIVELOUP			
JMP ENDLOOP	FRROR2: MOV	30h. #01h	// Zero Divisor (Y = 0) error		
			77 - 2.3 2.3301 (1 - 0) 61101		
end // Program ends		JAN LINDLOOF			
// Flogianienus	end		// Program ends		
	Cilu		// 110514111 €1143		