21BDS0340

Abhinav Dinesh Srivatsa

Microprocessors and Microcontrollers Lab

Lab Task – II

**Question 1**

Aim:

Write a program to transfer a string of data from code space starting at address 200H to RAM locations starting at 50H. The data is as shown below: 0200H: DB "Your Name and Register Number" e.g., Gordon Moore (12ABC3456)

Tools Required:

8051 microcontroller

Keil microcontroller software

Program:

|  |  |  |  |
| --- | --- | --- | --- |
| Memory Locations | Label | Mnemonics | Comments |
|  |  | ORG 0200H |  |
|  |  | DB "Abhinav Dinesh Srivatsa (21BDS0340)" | Creating data at 200H |
|  |  | ORG 0000H |  |
| 0000H |  | MOV DPTR, #200H | Moving to data start location |
| 0003H |  | MOV R0, #50H | Moving to transfer start location |
| 0005H |  | MOV R1, #23H | Need to loop 23H times, for 35 characters in string |
| 0007H |  | CLR A | Clearing accumulator |
| 0008H | LOOP: | MOVC A, @A+DPTR | Moving data to accumulator |
| 0009H |  | MOV @R0, A | Moving accumulators data to R0’s address |
| 000AH |  | INC DPTR | Increment DPTR for next space |
| 000BH |  | INC R0 | Increment R0 for next space |
| 000CH |  | CLR A | Clearing accumulator |
| 000D |  | DJNZ R1, LOOP | Decrement R0 and jump to LOOP if not zero |
|  |  | END |  |

Manual Calculations:

There are no manual calculations necessary to verify data transfer

Output Before/After:

Before:



After:

Result:

This program moves data from the ROM locations starting from 200H to the RAM locations starting from 50H

**Question 2**

Aim:

Write a program to add 10 bytes of data and store the result in registers R2 and R3. The bytes are stored in the ROM space starting at 200H

Tools Required:

8051 microcontroller

Keil microcontroller software

Program:

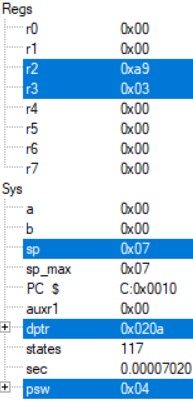
|  |  |  |  |
| --- | --- | --- | --- |
| Memory Locations | Label | Mnemonics | Comments |
|  |  | ORG 0200H |  |
|  |  | DB "wahAgb<idV" | Creating data at 200H |
|  |  | ORG 0000H |  |
| 0000H |  | MOV DPTR, #200H | Moving to data start location |
| 0003H |  | MOV R0, #10 | Need to loop 10 times for each number |
| 0005H |  | CLR A | Clearing accumulator |
| 0006H | LOOP: | MOVC A, @A+DPTR | Moving data to accumulator |
| 0007H |  | ADD A, R2 | Adding R2 to A |
| 0008H |  | JNC NADA | Jump to NADA if no carry |
| 000AH |  | INC R3 | Increment R3 if carry |
| 000BH | NADA: | MOV R2, A | Move data from A to R2 |
| 000CH |  | INC DPTR | Increment DPTR for next data |
| 000DH |  | CLR A | Clear accumulator |
| 000EH |  | DJNZ R0, LOOP | Decrement R0 and jump to LOOP if not zero |
|  |  | END |  |

Manual Calculations:

Text, letter

Description automatically generated

Output Before/After:

Chart

Description automatically generatedBefore: After:

Result:

This program allows for the addition of ten numbers stored starting from the location 200H and keeps the carry in R3 and sum in R2

**Question 3:**

Aim:

Write a program to get a byte of hex data from P1, convert it to unpacked BCD, and then to ASCII. For example, if P1 has 72H, after conversion (packed to unpacked BCD) we will have 37H and 32H. Place the ASCII result in RAM locations starting at 40H

Tools Required:

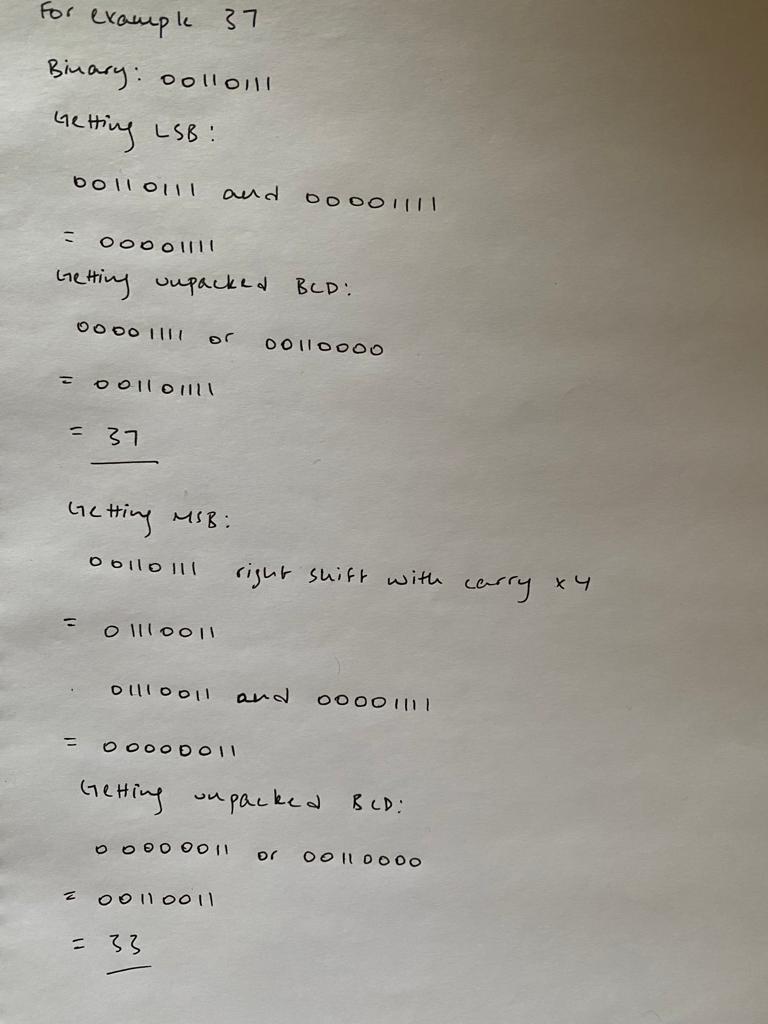
8051 microcontroller

Keil microcontroller software

Program:

|  |  |  |  |
| --- | --- | --- | --- |
| Memory Locations | Label | Mnemonics | Comments |
|  |  | ORG 0000H |  |
| 0000H |  | MOV A, P1 | Moving data from input P1 to accumulator |
| 0002H |  | MOV R1, A | Moving A to R1 to keep copy |
| 0003H |  | ANL A, #0FH | Getting LSB of accumulator |
| 0005H |  | ORL A, #30H | Unpacking BCD |
| 0007H |  | MOV R0, A | Moving unpacked BCD of the LSB to R0 |
| 0008H |  | MOV A, R1 | Reinitialising the value of accumulator |
| 0009H |  | RRC A | Right rotating with carry x4 to shift LSB and MSB |
| 000AH |  | RRC A |  |
| 000BH |  | RRC A |  |
| 000CH |  | RRC A |  |
| 000DH |  | ANL A, #0FH | Getting LSB of accumulator |
| 000FH |  | ORL A, #30H | Unpacking BCD |
| 0011H |  | MOV R1, A | Moving unpacked BCD of the LSB to R1 |
| 0012H |  | MOV 40H, R0 | Moving the unpacked LSB to 40H |
| 0014H |  | MOV 41H, R1 | Moving the unpacked MSB to 41H |
|  |  | END |  |

Manual Calculations:



Output Before/After:

Before:



After:



After (ASCII):



Result:

This program reads input from P1 and converts the packed BCD to unpacked BDC and stores in locations 40H and 41H

**Question 4**

Aim:

Write and assemble a program for the DS89C4x0 (90ns for one MC) chip to toggle all the bits of P0 and P1 continuously by sending 55H and AAH to these ports for every 1/4 of a second

Tools Required:

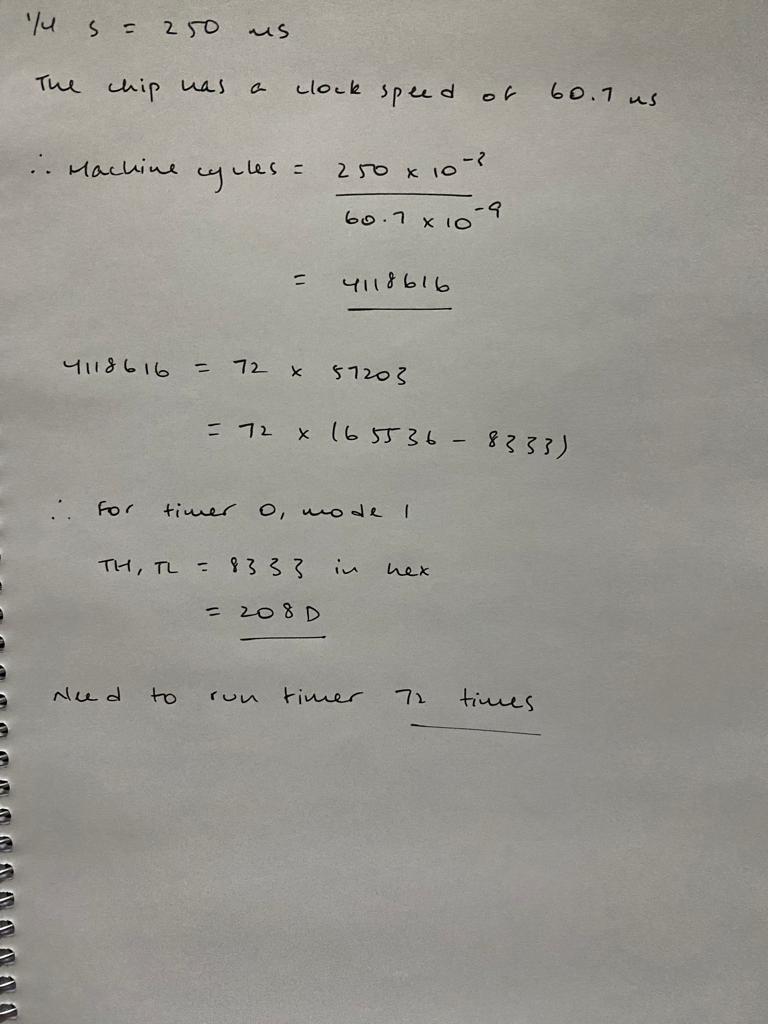
DS89C450 8051 microcontroller

Keil microcontroller software

Program:

|  |  |  |  |
| --- | --- | --- | --- |
| Memory Locations | Label | Mnemonics | Comments |
|  |  | ORG 0000H |  |
| 0000 |  | MOV TMOD, #01H | Setting TMOD for timer 0, mode 1 |
| 0003 |  | SETB P2.3 | Set signal on P2.3 bit |
| 0005 | QSEC: | MOV R0, #48H | Moving 72 to R0 to cycle 72 times |
| 0007 | LOOP: | MOV TL0, #8DH | Moving value 208DH, 8333 to initialise the timer |
| 000A |  | MOV TH0, #20H |  |
| 000D |  | SETB TR0 | Starting timer 0 |
| 000F | AGAIN: | JNB TF0, AGAIN | Remain here until flag 0 becomes 1 |
| 0012 |  | CLR TR0 | Clear timer 0 start |
| 0014 |  | CLR TF0 | Clear flag 0 |
| 0016 |  | DJNZ R0, LOOP | Decrement R0 and jump to LOOP if not zero |
| 0018 |  | CPL P2.3 | Compliment P2.3, changes signal value |
| 001A |  | SJMP QSEC | Jump to QSEC to restart timer |
|  |  | END |  |

Manual Calculations:



Output:

Chart, bar chart

Description automatically generated with medium confidence

Result:

This program creates a 50% duty cycle square wave that has a frequency of 0.25 milliseconds approximately

**Question 5**

Aim:  
Write and assemble a program to perform half adder and half subtractor using Keil simulator. Use acc.0 and acc.1 as input, acc.2 as sum output and acc.3 as carry output

Program:

|  |  |  |  |
| --- | --- | --- | --- |
| Memory Locations | Label | Mnemonics | Comments |
|  |  | ORG 0000H |  |
| 0000 |  | MOV A, #11B | Moving bits to create half adder |
| 0002 |  | MOV C, ACC.0 | Steps to find sum |
| 0004 |  | CPL ACC.1 |  |
| 0006 |  | ANL C, ACC.1 |  |
| 0008 |  | CPL ACC.1 |  |
| 000A |  | MOV ACC.2, C |  |
| 000C |  | MOV C, ACC.1 |  |
| 000E |  | CPL ACC.0 |  |
| 0010 |  | ANL C, ACC.0 |  |
| 0012 |  | CPL ACC.0 |  |
| 0014 |  | ORL C, ACC.2 |  |
| 0016 |  | MOV ACC.2, C | Moving sum to accumulator’s bit 2 |
| 0018 |  | MOV C, ACC.0 | Steps to find carry |
| 001A |  | ANL C, ACC.1 |  |
| 001C |  | MOV ACC.3, C | Moving carry to accumulator’s bit 3 |
|  |  | END |  |

Output Before/After:

Graphical user interface

Description automatically generated with medium confidenceChart

Description automatically generatedBefore: After:

Result:

This program allows to use the first two bits of the accumulator and perform the operations of a half adder and assign to the second two bits of the accumulator