**SUBJECT: MCA**

**PRACTICAL EXAMINATION EXPERIMENT LIST**

1. WAP to find largest number from a data block of 10 bytes.

ORG 0000H

MOV DPTR,#2500H

MOV R2,#0AH

MOV R1,#00H

UP: MOVX A,@DPTR

CJNE A, 01,NEXT

NEXT: JC DOWN

MOV R1,A

DOWN: INC DPTR

DJNZ R2,UP

MOV DPTR,#250AH

MOV A,R1

MOVX @DPTR,A

HERE: SJMP HERE

End

1. WAP to find smallest number from a data block of 10 bytes.

ORG 0000H

MOV DPTR,#2500H

MOV R2,#0AH

MOV R1,#0FFH

UP: MOVX A,@DPTR

CJNE A, 01,NEXT

NEXT: JNC DOWN

MOV R1,A

DOWN: INC DPTR

DJNZ R2,UP

MOV DPTR,#250AH

MOV A,R1

MOVX @DPTR,A

HERE: SJMP HERE

End

1. WAP to arrange series of ten 8-bit numbers in ascending order.

ORG 0000H

MOV R3,#0AH

LOOP: MOV R0,#30H

MOV R2,#0AH

LOOP1: MOV A,@R0

INC R0

MOV B,@R0

CJNE A,B,NEXT

NEXT: JC DOWN

MOV @R0,A

DEC R0

MOV @R0,B

INC R0

DOWN: DJNZ R2,LOOP1

DJNZ R3,LOOP

END

1. WAP to arrange series of ten 8-bit numbers in descending order.

ORG 0000H

MOV R3,#0AH

LOOP: MOV R0,#30H

MOV R2,#0AH

LOOP1: MOV A,@R0

INC R0

MOV B,@R0

CJNE A,B,NEXT

NEXT: JNC DOWN

MOV @R0,A

DEC R0

MOV @R0,B

INC R0

DOWN: DJNZ R2,LOOP1

DJNZ R3,LOOP

END

1. WAP to convert two digit BCD number into hex number.

mov r0,#30h

mov r1,#00h

mov a,@r0

anl a,#0fh

mov r1,a

mov a,@r0

anl a,#0f0h

swap a

mov b,a

mov a,#0ah

mul ab

add a,r1

inc r0

mov @r0,a

here: sjmp here

end

1. WAP to count number of 1’s in a given byte.

mov R0,#20H

mov R3,#08H

mov R2,#00H

Clr C

mov A,@R0

Up: RRC A

JNC down

INC R2

down: DJNZ R3,up

mov A,R2

end

1. WAP to count number of 0’s in a given byte.

mov R0,#20H

mov R3,#08H

mov R2,#00H

Clr C

mov A,@R0

Up: RRC A

JC down

INC R2

down: DJNZ R3,up

mov A,R2

mov 30H,A

end

1. WAP to count even number from a series of ten 8-bit numbers.

mov R0,#20H

mov R3,#0AH

mov R2,#00H

Clr C

up: mov A,@R0

RRC A

JC down

INC R2

down: INC R0

DJNZ R3,up

mov A,R2

mov 40H,A

End

1. WAP to count odd number from a series of ten 8-bit numbers.

mov R0,#20H

mov R3,#0AH

mov R2,#00H

Clr C

up: mov A,@R0

RRC A

JNC down

INC R2

down: INC R0

DJNZ R3,up

mov A,R2

mov 40H,A

End

1. WAP to exchange data blocks of 10 bytes.

mov R0,#30H

mov R1,#40H

mov R2 ,#0AH

up: mov A,@R0

XCH A,@R1

mov @R0,A

inc R0

inc R1

DJNZ R2,up

here: SJMP here

end

1. WAP to exchange data block of 10 bytes without exchange instruction.

ORG 0000H

MOV R2,#0AH

MOV R0,#30H

MOV R1,#40H //R3=TEMP

UP: MOV A,@R0

MOV R3,A

MOV A,@R1

MOV @R0,A

MOV A,R3

MOV @R1,A

INC R0

INC R1

DJNZ R2,UP

END

1. WAP to transfer data blocks of 10 bytes from internal memory to external memory.

ORG 0000H

MOV R2,#0AH //counter

MOV DPTR,#100H //external memory location\_destination

MOV R0,#30H //internal memory location\_source

UP: MOV A,@R0

MOVX @DPTR,A

INC R0

INC DPTR

DJNZ R2,UP

END

1. WAP to transfer data blocks of 10 bytes from internal memory to internal memory.

ORG 0000H

MOV R2,#0AH //counter

MOV R0,#20H //source

MOV R1,#80H //destination

UP: MOV A,@R0

MOV @R1,A

INC R0

INC R1

DJNZ R2,UP

END

1. WAP to transfer data blocks of 10 bytes from external memory to external memory.

ORG 0000H

MOV R2,#0AH //counter

MOV DPL,#00H //lower bits of addresses of 2400H & 2500H (i.e 00H)

MOV R3,#24H //source\_higher bits of 2400H (i.e 24H)

MOV R4,#25H //destination\_higher bits of 2500H (i.e 25H)

UP: MOV DPH,R3 //formation of address (2400H)

MOVX A,@DPTR

MOV DPH,R4 //formation of address (2400H)

MOVX @DPTR,A

INC DPL

DJNZ R2,UP

END

1. WAP to find 2’s complement of 16-bit number.

mov R0,#30H

mov R1,#31H

mov A,@R0

CPL A

ADD A,#01H

mov @R0,A

mov A,@R1

CPL A

ADDC A,#00H

mov @R1,A

end

1. WAP to multiply 16-bit no. into 8-bit number.

mov r0,#20h // lb of 16b in 20

mov a,@r0

mov r0,#22h //8b in 22

mov b,@r0

mul ab

mov 23h,a //lb1 in 23

mov r1,b //hb1 in r1

mov b,@r0 //hb of 16b in 21

mov r0,#21h

mov a,@r0

mul ab

add a,r1 //hb1+lb2

mov r1,#24h //hb1+lb2 in 24

mov @r1,a

mov a,b

addc a,#00h

mov r0,#25h

mov @r0,a //hb2+carry in 25

here: sjmp here

end

1. WAP to evaluate expression ‘x2 + x + 1’ where ‘x’ is an 8-bit number.

org 00h

mov r0,#10h

mov r1,#20h

mov a,@r0

mov b,a

mul ab

add a,@r0

mov @r1,a

mov a,b

addc a,#00h

mov b,a

mov a,@r1

add a,#1h

inc r0

mov @r0,a

mov a,b

addc a,#00h

inc r0

mov @r0,a

stop: sjmp stop

end

1. WAP to convert packed BCD into unpacked BCD number.

MOV R0,#30H

MOV A,@R0

MOV B,A

ANL A,#0F0H

SWAP A

MOV 31H,A

MOV A,B

ANL A ,#0FH

MOV 32H,A

END

1. WAP to convert unpacked BCD into packed BCD number.

MOV R0,#30H

MOV R1,#31H

MOV A,@R0

MOV B,@R1

RL A

RL A

RL A

RL A

ADD A,B

MOV 32H,A

END

1. WAP to add two 4-digit BCD numbers.

mov r0,#10h

mov dph,@r0

inc r0

mov dpl,@r0

inc r0

mov b,@r0

inc r0

mov a,@r0

add a,dpl

da a

mov 16h,a

mov a,b

addc a,dph

da a

mov 15h,a

mov a,#00h

addc a,#00h

mov 14h,a

stop: sjmp stop

end

1. WAP to subtract 2-digit BCD numbers.

mov r0,#10h

mov a,@r0

inc r0

subb a,@r0

da a

inc r0

mov @r0,a

end

1. WAP to find sum of data block of 10-bytes.

mov a,#00h

mov r0,#30h

mov r2,#00h

mov r3,#0ah

up: add a,@r0

jnc down

inc r2

down: inc r0

djnz r3,up

mov r1,#3ah

mov @r1,a

inc r1

mov a,r2

mov @r1,a

here: sjmp here

end

1. WAP to perform 8-bit division.

MOV DPTR,#3000H

MOVX A,@DPTR

MOV R0,A

INC DPTR

MOVX A,@DPTR

MOV B,A

MOV A, R0

DIV AB

MOV DPTR,#3020H

MOVX @DPTR,A

INC DPTR

MOV A,B

MOVX @DPTR,A

END

1. WAP to convert packed BCD into ASCII code.

org 00h

mov r0,#20h

mov a,@r0

anl a,#0f0h

swap a

add a,#30h

mov 21h,a

mov a,@r0

anl a,#0fh

add a,#30h

mov 22h,a

stop: sjmp stop

end

|  |
| --- |
| 26. ARM programs) |
|  |  |
|  | a. Add memory block of 10 32-bit numbers |
|  |  |
|  | area add, code, readonly |
|  | entry |
|  | mov r0, #0 |
|  | mov r1, #0x0A |
|  | mov r4, #0 |
|  | adr r2,data |
|  | loop ldr r3, [r2,r4] |
|  | add r0, r3, r0 |
|  | add r4, r4, #4 |
|  | subs r1, r1, #1 |
|  | bne loop |
|  | stop b stop |
|  |  |
|  | Align |
|  | data dcd 1,1,1,1,1,1,1,1,1,1 |
|  | End |
|  |  |
|  | b. Perform Math operations - add, sub, mul |
|  |  |
|  | area maths, code, readonly |
|  | Entry |
|  | mov r7,#0x40000000 |
|  | mov r8,#0x40000004 |
|  |  |
|  | ldr r0,[r7] |
|  | ldr r1,[r8] |
|  | add r2,r0,r1 |
|  | str r2,[r7,#8] |
|  |  |
|  | sub r2,r2,r0 |
|  | str r2,[r7,#12] |
|  |  |
|  | umull r3,r2,r1,r0 |
|  | str r3,[r7,#16] |
|  | str r2,[r7,#20] |
|  | exit b exit |
|  | End |
|  |  |
|  |  |
|  | c. Smallest Number using ARM-7 |
|  |  |
|  | area smallest,code,readonly |
|  | entry |
|  | mov r0,#0x00 |
|  | mov r1,#0x0a |
|  | adr r2,array |
|  | mov r3,#0 |
|  | loop ldr r4,[r2,r3] |
|  | cmp r0,r4 |
|  | bcc carry |
|  | carr2 add r3,r3,#4 |
|  | subs r1,r1,#1 |
|  | bne loop |
|  | stop b stop |
|  | carry mov r0,r4 |
|  | b carr2 |
|  | Align |
|  | array dcd 2,4,9,8,4,7,3,2,1,0 |
|  | End |
|  |  |
|  | d. Largest Number using ARM-7 |
|  |  |
|  | area largest,code,readonly |
|  | entry |
|  | mov r0,#0x40000000 |
|  | mov r1,#0x0 |
|  | mov r2,#0x0A |
|  | up ldr r3,[r0] |
|  | cmp r1,r3 |
|  | bcs down |
|  | mov r1,r3 |
|  | down add r0,r0, #0x4 |
|  | subs r2,r2,#0x1 |
|  | bne up |
|  | mov r4,#0x40000038 |
|  | str r1,[r4] |
|  | here b here |
|  | end |