Facit/Lösningsförslag till Dugga i Mikrodatorteknik 4p

Moment B, Assemblerprogrammering

1 Statusflaggors beteende (20p)

Vad blir C-, N- och Z- flaggornas värden samt D0:s värden efter varje instruktion i följande program. (redovisa beräkningar).

Not	Program	D0	C	N	Z
	MOVE.B #$7E,D0	0111 1110	0	0	0
1	ANDI.B #$81,D0	0000 0000	0	0	1
2	ORI.B #$8F,D0	1000 1111	0	1	0
3	ADDI.B #$E1,D0	0111 0000	1	0	0

Eftersom endast de 8 minst signifikanta bitarna påverkas av varje instruktion så redovisas innehållet endast i den minst signifikanta byten av D0. Värdet av övriga 24 av totalt 32 bitar är okänt.

1)

```
     7E =   0111 1110
AND  81 =   1000 0001
            0000 0000
```

2)

```
     00 =   0000 0000
OR   8F =   1000 1111
            1000 1111
```

3)

```
      1
      8F
   +  E1
     170
```

ettan i den mest signifikanta positionen innebär att vi fått en carry när vi summerat två bytes.

2 Adresseringsmetoder (20p)

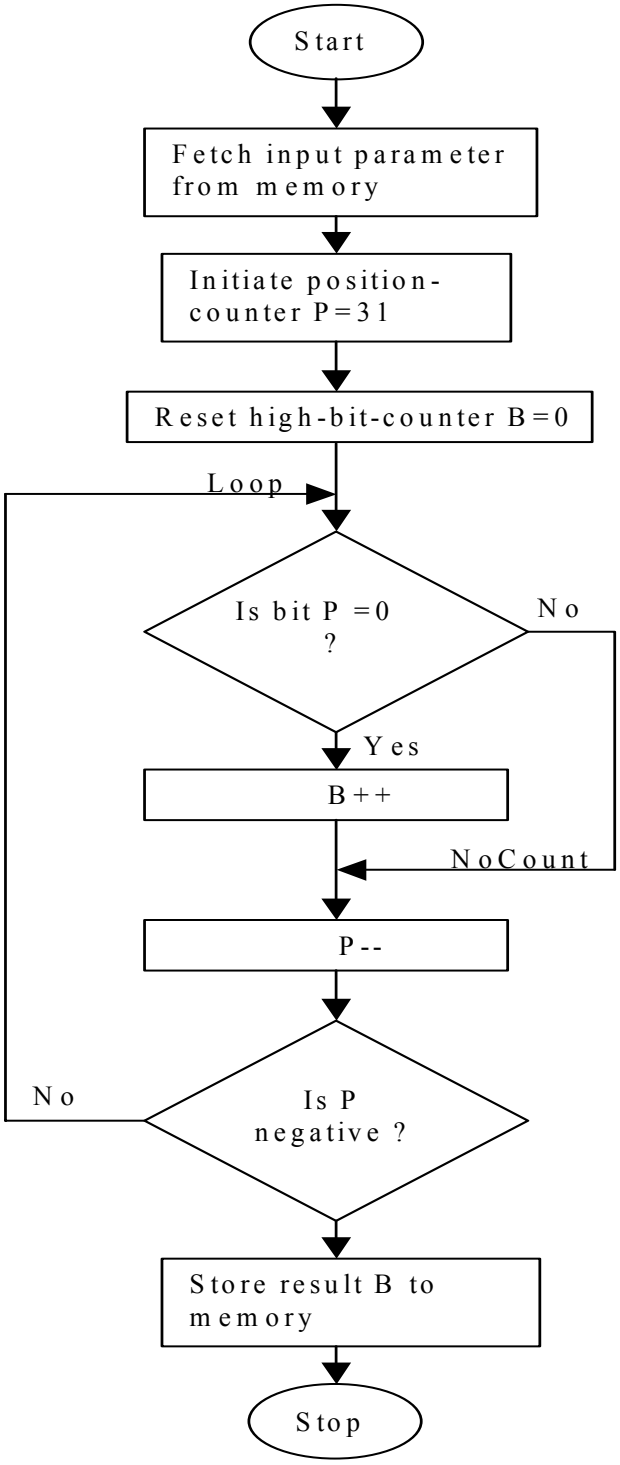
Adress (Hexadecimal)	Data i minnet (Hex)
4000	39
4001	45
4002	A6
4003	F6
4004	D4
4005	AA
4006	67
4007	89
4008	34
4009	22
400A	00
400B	23
400C	16
400D	24
400E	90
400F	87

D3 = $00000000 , A0= $00004004 innan instruktionen

Instruktion	D3	A0
MOVE.L $4000, D3	3945A6F6	00004004
MOVE.B 4(A0), D3	00000034	00004004
MOVE.W -(A0), D3	0000A6F6	00004002
MOVE.L (A0)+, D3	D4AA6789	00004008

3 Räkna antalet nollor i ett ord (40p)

Skriv ett program som räknar antalet noll-värda bit-positioner (bitpositioner = 0) i ett 32-bitars ord. Ordet som skall analyseras finns på adresserna FF0000- FF0003 $_{HEX}$. Resultatet av analysen, dvs antalet nollor skall sparas till adressen FF0004 $_{HEX}$. Programkoden skall starta f.o.m. adressen $4000_{HEX.}$ Redovisa algoritmen med flödesdiagram eller text.



```
                ORG         $FF0000

Param:                      DS.L        1
NumOfLowBits:               DS.B        1


                ORG         $4000

                MOVE.B      #31,D0              * Initiated to bit 32
                MOVE.L      Param,D1            * Get parameter into D1
                MOVE.B      #0,NumOfLowBits     * Reset the low bit counter
Loop:
                BTST.L      D0,D1
                BNE         NoCount
                ADDI.B      #1,NumOfLowBits     * Increment the low bit counter
NoCount:
                SUBI.B      #1,D0               * decrement to next bit
                BPL         Loop                * continue to count until bit 0
```

4 Parameteröverföring (20p)

Visa med programkod hur en 16-bitars parameter kan överföras genom värde till en subrutin via stacken. Visa hur parametern hanteras i det anropande programmet och hur subrutinen läser parameterns värde. Rita en bild över hur stacken ser ut efter anropet till subrutinen men före återhoppet till det anropande programmet. Endast programkod för parameteröverföringen behöver skrivas!

Segment of calling program

```
                .
                .
                .
MOVE.W    #$FFFF, -(SP)      *Push parameter = FFFF onto the stack
BSR       MySubroutine
ADDQ.L    #2,SP              *Remove parameter from the stack

                .
                .
                .
```

Subroutine

MySubroutine:

```
MOVE.W    4(SP),D0           *Load parameter into D0.W

                .
                .
                .
RTS
```