Homework 12 Solutions

1. Here is a short program that shows all addressing modes:

We are given a table of student's test scores where there are three scores in a semester per student. Unfortunately, the person who entered the grades put them in the wrong order. They presently are in the following order. Student #1 test #2/test #3/test #1, Student #2 test #2/test #3/test #1… Student #200 test #2/test #3/test #1. We would like them to be in the following order. Student #1 test #1/test #2/test #3, Student #2 test #1/test #2/test #3… Student #200 test #1/test #2/test #3. Assume all scores start at 1000H (SRAM).

```
N         EQU     200
TABLE     EQU     $1000
NEG1      EQU     %11111111  ; $FF

          ORG     $1800      ; temporary data area
TEMP      DS.B    1
COUNT     DS.B    1

          ORG     $0
          LDAA    #N         ; (immediate addressing)
          STAA    COUNT      ; (extended addressing)
          LDX     #TABLE     ; ptr to top of table (immediate addressing)
Loop_pt   LDAB    0,X        ; get test #2 score (indexed addressing)
          LDAA    1,X        ; get test #3 score (indexed addressing)
          STAA    TEMP       ; save test #3 score in temp area (extended)
          LDAA    2,X        ; get test #1 score (indexed)
Re_order_data
          STAA    0,X        ; store test #1 score (indexed)
          STAB    1,X        ; store test #2 score (indexed)
          LDAA    TEMP       ; get test #3 score (extended)
          STAA    2,X        ; store test #3 score (indexed)
Check_counter
          LDAA    COUNT      ; retrieve count (extended)
          LDAB    #NEG1      ; decrement counter (immediate)
          SUM_BA             ; count = count -1 (inherent)
          BEQ     END        ; if count = 0 then end (branch addressing)
          STAA    COUNT      ; save count (extended)
          INX                ; inc ptr (inherent)
          INX                ; inc ptr (inherent)
          INX                ; inc ptr (inherent)
          BNE     Loop_pt    ; if count != 0 then loop (branch addressing)
END       BEQ     END        ; something to do (branch addressing)
```

Homework 12 Solutions

2. a) Here is the hand assembly (shown in two different formats)

Address (Hex)	Data (Hex)	*Instructions*		
0	08		*LDX*	*#$1100*
1	00			
2	11			
3	02		*LDAA*	*#$10*
4	10			
5	06		*STAA*	*$1200*
6	00			
7	12			
8	0C	*LOOP:*	*LDAA*	*0,X*
9	00			
A	0E		*LDAB*	*$10,X*
B	10			
C	15		*SUM_AB*	
D	1F		*SHFB_R*	
E	12		*STAB*	*$20,X*
F	20			
10	30		*INX*	
11	03		*LDAB*	*#$FF*
12	FF			
13	04		*LDAA*	*$1200*
14	00			
15	12			
16	14		*SUM_BA*	
17	20		*BEQ*	*DONE*
18	1E			
19	06		*STAA*	*$1200*
1A	00			
1B	12			
1C	21		*BNE*	*LOOP*
1D	08			
1E	20	*DONE:*	*BEQ*	*DONE*
1F	1E			

Address (Hex)	Data (Hex)	*Instructions*		
0	08 00 11		*LDX*	*#$1100*
3	02 10		*LDAA*	*#$10*
5	06 00 12		*STAA*	*$1200*
8	0C 00	*LOOP:*	*LDAA*	*0,X*
A	0E 10		*LDAB*	*$10,X*
C	15		*SUM_AB*	
D	1F		*SHFB_R*	
E	12 20		*STAB*	*$20,X*
10	30		*INX*	
11	03 FF		*LDAB*	*#$FF*
13	04 00 12		*LDAA*	*$1200*
16	14		*SUM_BA*	
17	20 1E		*BEQ*	*DONE*
19	06 00 12		*STAA*	*$1200*
1C	21 08		*BNE*	*LOOP*
1E	20 1E	*DONE:*	*BEQ*	*DONE*

This code grabs a number from Table #1 @ $1100 and a number from Table #2 @ $1110 and then computes an average value which is then stored in a Table #3, starting at $1120.

b) This process is repeated 16 ($10) decimal times.
c) We must save the count in a temporary memory location because the A register is corrupted inside the loop that retrieves the data and computes the average value.

Homework 12 Solutions

2. d) Re-written code.

```
*************************************
Num     EQU     $10
Numx2   EQU     NUM*2   ; $20
Neg1    EQU     $FF     ; 2's complement minus 1
* Could replace the above line with a
*       Minus1  DC.B    $FF
*   If did this replacement, would also need to change code.
*   The line LDAB #Neg1 would be replaced by
*       LDAB    Minus1  .
*************************************
        ORG     $1200
Count   DS.B    1

        ORG     $1100
Table   DS.B    Num     ; for Table #1
        DS.B    Num     ; for Table #2
        DS.B    Num     ; for Table #3
*************************************
* Main program

        ORG     $0      ; assembler directive (origin) to tell where code will be placed in memory
        LDX     #Table  ; pointer to data
        LDAA    #Num    ; counter value
        STAA    Count   ; counter will be saved in memory to free up a CPU register
LOOP:   LDAA    0,X     ; get 1st data value
        LDAB    Num,X   ; get 2nd data value
        SUM_AB          ; data1 + data2
        SHFB_R          ; divide sum by 2
        STAB    Numx2,X ; store average. value
        INX             ; increment pointer
        LDAB    #Neg1   ; -1 in 2's complement format
        LDAA    Count   ; count = count – 1
        SUM_BA
        BEQ     DONE    ; branch to done if count is zero
        STAA    Count   ; else, save counter value
        BNE     LOOP    ; and repeat loop
DONE:   BEQ     DONE    ; loop forever
```

Homework 12 Solutions

3. Here is the program that corresponds to the machine code listed in problem #3:

Addr	Op Codes	Instructions	Comments
		ORG $100	
$100	06 00 14	STAA $1400	; temporarily store the A reg value
$103	1A	COMA	; /A
$104	16	AND_BA	; /A*B
$105	06 01 14	STAA $1401	; save /A*B for future use
$108	04 00 14	LDAA $1400	; restore original A reg value
$10B	1B	COMB	; /B
$10C	16	AND_BA	; A */B
$10D	00	TAB	
$10E	04 01 14	LDAA $1401	; retrieve /A*B
$111	18	OR_BA	; (A*/B) + (/A*B)

```
                ORG         $100
$100    06 00 14 STAA       $1400    ; temporarily store the A reg value
$103    1A       COMA                ; /A
$104    16       AND_BA              ; /A*B
$105    06 01 14 STAA       $1401    ; save /A*B for future use
$108    04 00 14 LDAA       $1400    ; restore original A reg value
$10B    1B       COMB                ; /B
$10C    16       AND_BA              ; A */B
$10D    00       TAB
$10E    04 01 14 LDAA       $1401    ; retrieve /A*B
$111    18       OR_BA       ; (A*/B) + (/A*B)
```

This code performs the exclusive or of registers A and B (A xor B = A*/B + /A*B) and then returns this value in register A. Two temporary locations are required, 1400H and 1401H.

4. Code to count the number of (decimal) 37's in memory from $1000-$107F:

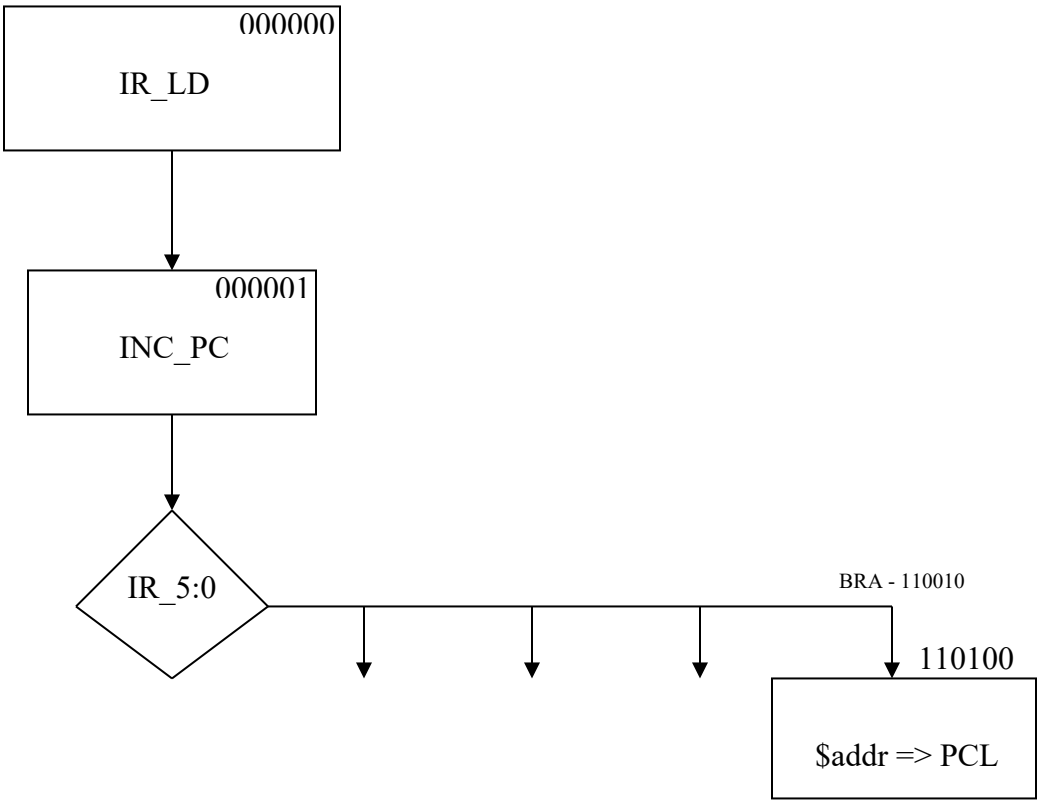
```
*********************************
* The final result is in Num37s and is also passed to the A register.
*********************************
NUM       EQU       $7F+1       ; = $80
Neg1      EQU       $FF
Neg37     EQU       $DB         ; -37 = -$25 = -(%0010 0101)
Table     EQU       $1000
*                                ;   -(%0010 0101) => %1101 1010 + %1 = %1101 1011 = $DB
          ORG       $1100       ; The below program uses these temporary locations that are reserved in memory
Num37s    DS.B      1           ; Keep track of the number of 37s found
LoopCnt   DS.B      1           ; Loop counter
* Neg37    DS.B      1           ; Holds -37 value used to test if the current data is 37
          ORG       $0
          LDAA      #0
          STAA      Num37s      ; Initialize the count of 37's found to zero
          LDAA      #NUM        ; Initialize the loop counter (how many times the loop is executed)
          STAA      LoopCnt     ;    ...
          LDX       #Table      ; Initialize table pointer to point to the beginning of the table
TOP       LDAA      0,X         ; Get first table value
          LDAB      #Neg37      ; Get -37
          SUM_BA                ; Add table value to –37.  If result is zero, table value was 37.
          BNE       SKIP_INC    ;    If result is not zero, the table value was not 37, so don't increment.
```

Homework 12 Solutions

```
            LDAA    Num37s      ; increment the number of $37s count value
            LDAB    #1
            SUM_AB
            STAB    Num37s
SKIP_INC    LDAA    LoopCnt
            LDAB    #Neg1       ; Decrement loop counter
            SUM_BA              ;   ...
            BEQ     DONE
            STAA    LoopCnt
            INX                 ; Increment the data pointer
            BNE     TOP
DONE        LDAA    Num37s
WAIT        BNE     WAIT
            BEQ     WAIT
```

Homework 12 Solutions

5. BRA $addr



Q5:0	IR5:0	Z	N	D5:0	MSA	MSB	MSC	IR_LD	RW	PMXY	SEL	PC_LD	M_LD	X_LD	Y_LD	XD	YD
000001	110010	-	-	110100	01	10	0000	0	1	1000	00	00	00	00	00	0	0
110100	------	-	-	000000	01	10	0000	0	1	0000	00	10	00	00	00	0	0