DSP Lab Submission, Madhav Lekkala (2020EEP2489)

26 November 2021 01:00

1. Develop a dspic executable program that will generate a series 2, 5, 8, 11,14,17,... and store in the memory.

Code :-

```
......
         Submission by : Madhav Lekkala
3
           Roll Number
                       : 2020EEP2489
                      : Problem 1
          Problem #
           Program des
5
                       : Generate series 2,5,8,11,14
     ......
     .equ __30F6010, 1
9
           .include "p30f6010.inc"
10
           .global __reset
12
           .text
13
                             ; MAIN FUNCTION START
       MOV #0x1000, W4
                             ; LOAD START ADDRESS TO W4
14
15
       MOV #0x0001, W0
                             ; PROGRAM COUNTER (SELF)
16
        MOV #2, W1
                             ; STARTING VALUE OF 2 STORED IN W1
17
        LOOP1: DO #14, END1
                            ; RUN LOOP FOR 15 TIMES
19
        MOV W1, [W4]
                             ; MOV UPDATED VALYE TO W4 ADDRESS
20
        INC2 W4, W4
                              ; INCREMENT ADDRESS BY 0x02
        ADD #3, W1
                             ; ADD 3 TO PREV VALUE
22
        END1: INC WO, WO
                            ; INCREMENT SELF PROGRAM COUNTER
23
```

Storage starts from 0×1000
Before Running vode >

					_				
Address	00	02	04	06	08	0A	0C	0E	ASCII
0FE0	0000	0000	0000	0000	0000	0000	0000	0000	
OFF0	0000	0000	0000	0000	0000	0000	0000	0000	
1000	0000	0000	0000	0000	0000	0000	0000	0000	
1010	0000	0000	0000	0000	0000	0000	0000	0000	
1020	0000	0000	0000	0000	0000	0000	0000	0000	
1030	0000	0000	0000	0000	0000	0000	0000	0000	
1040	0000	0000	0000	0000	0000	0000	0000	0000	
1050	0000	0000	0000	0000	0000	0000	0000	0000	
1060	0000	0000	0000	0000	0000	0000	0000	0000	
1070	0000	0000	0000	0000	0000	0000	0000	0000	
1080	0000	0000	0000	0000	0000	0000	0000	0000	
1090	0000	0000	0000	0000	0000	0000	0000	0000	
10A0	0000	0000	0000	0000	0000	0000	0000	0000	
10B0	0000	0000	0000	0000	0000	0000	0000	0000	
10C0	0000	0000	0000	0000	0000	0000	0000	0000	
10D0	0000	0000	0000	0000	0000	0000	0000	0000	
10E0	0000	0000	0000	0000	0000	0000	0000	0000	
10F0	0000	0000	0000	0000	0000	0000	0000	0000	
1100	0000	0000	0000	0000	0000	0000	0000	0000	
1110	0000	0000	0000	0000	0000	0000	0000	0000	

After Running code

Address	00	02	04	06	08	0A	0C	0E	ASCII
OFE0	0000	0000	0000	0000	0000	0000	0000	0000	
OFF0	0000	0000	0000	0000	0000	0000	0000	0000	
1000	0002	0005	8000	000B	000E	0011	0014	0017	
1010	001A	001D	0020	0023	0026	0029	002C	0000	#. &.).,
1020	0000	0000	0000	0000	0000	0000	0000	0000	
1030	0000	0000	0000	0000	0000	0000	0000	0000	
1040	0000	0000	0000	0000	0000	0000	0000	0000	
1050	0000	0000	0000	0000	0000	0000	0000	0000	
1060	0000	0000	0000	0000	0000	0000	0000	0000	
1070	0000	0000	0000	0000	0000	0000	0000	0000	
1080	0000	0000	0000	0000	0000	0000	0000	0000	
1090	0000	0000	0000	0000	0000	0000	0000	0000	
10A0	0000	0000	0000	0000	0000	0000	0000	0000	
10B0	0000	0000	0000	0000	0000	0000	0000	0000	
10C0	0000	0000	0000	0000	0000	0000	0000	0000	
10D0	0000	0000	0000	0000	0000	0000	0000	0000	
10E0	0000	0000	0000	0000	0000	0000	0000	0000	
10F0	0000	0000	0000	0000	0000	0000	0000	0000	
1100	0000	0000	0000	0000	0000	0000	0000	0000	
1110	0000	0000	0000	0000	0000	0000	0000	0000	
1120	0000	0000	0000	0000	0000	0000	0000	0000	
1130	0000	0000	0000	0000	0000	0000	0000	0000	
1140	0000	0000	0000	0000	0000	0000	0000	0000	

The Series is Stored accordingly in Hex

Working Repisters

Address /	Name	Hex	Decimal	Binary					Char
0000	WREG	0x0000	0	00000000	00000000				11
0002	WREG	l 0x0000	0	00000000	00000000				''
0004	WREG	2 0x0000	0	00000000	00000000				''
0006	WREG	3 0x0000	0	00000000	00000000				11
8000	WREG	4 0x0000	0	00000000	00000000				11
000A	WREG	5 0x0000	0	00000000	00000000				''
000C	WREG	6 0x0000	0	00000000	00000000				''
000E	WREG	7 0x0000	0	00000000	00000000				''
0010	WREG	0x0000	0	00000000	00000000				''
0012	WREG	9 0x0000	0	00000000	00000000				''
0014	WREG	10 0x0000	0	00000000	00000000				''
0016	WREG	11 0x0000	0	00000000	00000000				''
0018	WREG	12 0x0000	0	00000000	00000000				''
001A	WREG	13 0x0000	0	00000000	00000000				''
001C	WREG	14 0x0000	0	00000000	00000000				''
001E	WREG	15 0x0000	0	00000000	00000000				''
0020	SPLI	00000x0 P	0	00000000	00000000				''
0022	ACCA	0x0000000000	0	00000000	00000000	00000000	00000000	00000000	''
0022	ACCA	L 0x0000	0	00000000	00000000				''
0024	ACCA	0x0000	0	00000000	00000000				''
0026	ACCA	J 0x0000	0	00000000	00000000				''
0028	ACCB	0x0000000000	0	00000000	00000000	00000000	00000000	00000000	''
0028	∆CCB:	. 0x0000	0	00000000	00000000				1 1

Before

Address /	Name	Hex	Decimal	Binary	Char
0000	WREG0	0x0010	16	00000000 00010000	11
0002	WREG1	0x002F	47	00000000 00101111	'./'
0004	WREG2	0x0000	0	00000000 00000000	''
0006	WREG3	0x0000	0	00000000 00000000	''
8000	WREG4	0x101E	4126	00010000 00011110	11
000A	WREG5	0x0000	0	00000000 00000000	''
000C	WREG6	0x0000	0	00000000 00000000	''
000E	WREG7	0x0000	0	00000000 00000000	''
0010	WREG8	0x0000	0	00000000 00000000	''
0012	WREG9	0x0000	0	00000000 00000000	''
0014	WREG10	0x0000	0	00000000 00000000	''
0016	WREG11	0x0000	0	00000000 00000000	''
0018	WREG12	0x0000	0	00000000 00000000	''
001A	WREG13	0x0000	0	00000000 00000000	''
001C	WREG14	0x0000	0	00000000 00000000	''

After

2. Develop a dspic executable program that will sort the given series in ascending/descending order. (The series is:1, 34, 5,21,1, 2, 13, 3,55,89,8)

Code

```
.equ __30F6010, 1
 3
             .include "p30f6010.inc"
             .global __reset
 5
           reset:
          MOV #1, WO
8
 9
          MOV W0, 0x1000
          MOV #34, WO
10
11
          MOV W0, 0x1002
          MOV #5, WO
12
                                         storing the series
from 0x1000
13
          MOV W0, 0x1004
14
          MOV #21, W0
         MOV W0, 0x1006
15
16
          MOV #1, W0
17
          MOV W0, 0x1008
18
         MOV #2, W0,
19
          MOV W0, 0x100A
20
          MOV #13, W0
          MOV WO, 0x100C
22
          MOV #3, WO
          MOV WO, 0x100E
23
24
          MOV #55, W0
25
          MOV W0, 0x1010
          MOV #89, WO
26
27
          MOV W0, 0x1012
28
          MOV #8, WO
29
          MOV W0, 0x1014
30
31
32
          MOV #0x0000. WO
                           ; PROGRAM COUNTER FOR INNER LOOP
          MOV #0x0000, W5
                                  ; PROGRAM COUNTER FOR OUTER LOOP
33
34
35
          LOOP4: DO #9, END4
                                  ; OUTER LOOP FOR BUBBLE SORT
36
          MOV #0x1000, W1
                                  ; STARTING ADDRESS TO W1
                                  ; STARTING ADDRESS TO W2
          MOV #0x1002, W2
37
38
           LOOP1: DO #9, END1 ; INEER LOOP FOR IMMEDIATE SORT
                                                                                       Bubble
sort
code
                                 ; TEMP STORAGE OF VALUE 1 FOR CP
39
              MOV [W1], W3
                                  ; TEMP STORAGE OF VALUE 2 FOR CP
              MOV [W2], W4
40
41
               CP.B W3, W4
                                  ; CP SETS CARRY IF W3>W4
                                  ; BRANCH TO LOOP2 IF CARRY IS SET
               BRA C. LOOP2
42
                                  ; BRANCH TO LOOP3 IS IT ISNT
43
               BRA LOOP3
44
45
              LOOP2: MOV W3, [W2] ; SWAPPING NUMBERS IF W3>W4
46
               MOV W4, [W1]
                                      ; SWAPPING NUMBERS IF W3>W4
                                      ; BRANCH TO LOOP3 (UNCONDITIONAL)
               BRA LOOP3
47
48
49
               LOOP3: INC2 W1, W1
                                     ; INCREMENT ADDRESS 1 FOR NEXT SWEEP
                                     ; INCREMENT ADDRESS 2 FOR NEXT SWEEP
50
               INC2 W2, W2
51
           END1: INC WO, WO
                                      ; INNER LOOP PC INCREMENT
          END4: INC W5, W5
                                      ; OUTER LOOP PC INCREMENT
52
53
54
55
```

	Addre	SS	00	02	04	06	0	8	0A	0C	0E	ASCII		
	0FC0	(0000	0000	0000	000	00 00	00	0000	0000	0000			
	0FD0	(0000	0000	0000	000	0 00	00	0000	0000	0000			
1	0FE0	(0000	0000	0000	-	_		0000		0000			
,	OFF0		0000	0000			_		0000		_			
	1000		0001	0022	0005				0002	_	_			0.1.
	1010		0037	0059					0000	_	_		•••••	Before
	1020		0000	0000		_	_		0000					,
	1030		0000	0000					0000					
	1040			0000	-	-			0000		_			
	1050		0000	0000					0000					
	1060		0000	0000					0000					
	1070		0000	0000	0000		-		0000					
	1080	- (0000	0000	0000	000	00	00	0000	0000	0000			
C)	0000	000	00 00	00 0	000	0000	00	00 0	0000	0000			
DI		0000	_				0000							
E (0000	-				0000	-	00 0		0000			-
E(0000					0000	-	00 0		0000			-
0 (0000					0005	-			0015			-
11		0001					0000				0000	".7.Y		AHW
21		0000					0000	_			0000			117701
31		0000					0000	-	00 0		0000	<i>E</i>		-
11		0000	_				0000	_			0000		·····	-
51		0000					0000	-			0000		$\overline{}$	-
61		0000					0000							ال مالات
71								_	00 0		0000			Ascendin order
		0000					0000	_	00 0		0000			A
_	J	0000	_		-		0000	_			0000			AS Cender
9 (10000												

If you want to stree in Descending order -s

Change hise 43 to -> CP.B W4, W3

3. Develop a dspic executable program that will identify the minimum/maximum values in the given series and stores in memory location. (The series is:1, 34, 5,21,1, 2, 13, 3,55,89,8)

We cen use problem 2 vode here Code.

```
MOV #0x0000, W0
                                   ; PROGRAM COUNTER FOR INNER LOOP
33
           MOV #0x0000, W5
                                   ; PROGRAM COUNTER FOR OUTER LOOP
34
35
           MOV #0x1000, W1
                                   ; STARTING ADDRESS TO W1
                                  ; STARTING ADDRESS TO W2
           MOV #0x1002, W2
                                 ; INEER LOOP FOR IMMEDIATE SORT
; TEMP STORAGE OF VALUE 1 FOR CP
37
           LOOP1: DO #9, END1
38
              MOV [W1], W3
                                  ; TEMP STORAGE OF VALUE 2 FOR CP
               MOV [W2], W4
39
40
               CP.B W3, W4
                                   ; CP SETS CARRY IF W3>W4
41
               BRA C, LOOP2
                                  ; BRANCH TO LOOP2 IF CARRY IS SET
                                   ; BRANCH TO LOOP3 IS IT ISNT
43
44
               LOOP2: MOV W3, [W2] ; SWAPPING NUMBERS IF W3>W4
45
               MOV W4, [W1]
                                       ; SWAPPING NUMBERS IF W3>W4
46
               BRA LOOP3
                                      ; BRANCH TO LOOP3 (UNCONDITIONAL)
47
48
               LOOP3: INC2 W1, W1
                                      ; INCREMENT ADDRESS 1 FOR NEXT SWEEP
                                       ; INCREMENT ADDRESS 2 FOR NEXT SWEEP
               INC2 W2, W2
49
50
           END1: INC WO, WO
                                       ; INNER LOOP PC INCREMENT
51
52
           ; MAXIMUM VALUE IS STORED IN 0x1014 address (last value of array)
53
54
```

						•			
Address	00	02	04	06	08	0A	0C	0E	ASCII
0FC0	0000	0000	0000	0000	0000	0000	0000	0000	
OFD0	0000	0000	0000	0000	0000	0000	0000	0000	
OFE0	0000	0000	0000	0000	0000	0000	0000	0000	
OFF0	0000	0000	0000	0000	0000	0000	0000	0000	
1000	0001	0022	0005	0015	0001	0002	000D	0003	"
1010	0037	0059	8000	0000	0000	0000	0000	0000	7.Y
1020	0000	0000	0000	0000	0000	0000	0000	0000	
1030	0000	0000	0000	0000	0000	0000	0000	0000	
1040	0000	0000	0000	0000	0000	0000	0000	0000	
1050	0000	0000	0000	0000	0000	0000	0000	0000	
1060	0000	0000	0000	0000	0000	0000	0000	0000	
1070	0000	0000	0000	0000	0000	0000	0000	0000	
1080	0000	0000	0000	0000	0000	0000	0000	0000	
	0FC0 0FD0 0FE0 0FF0 1000 1010 1020 1030 1040 1050 1060	0FC0 0000 0FD0 0000 0FE0 0000 0FF0 0000 1000 0001 1010 0037 1020 0000 1030 0000 1040 0000 1050 0000 1060 0000	OFCO 0000 0000 OFDO 0000 0000 OFEO 0000 0000 OFFO 0000 0000 1000 0001 0022 1010 0037 0059 1020 0000 0000 1030 0000 0000 1040 0000 0000 1050 0000 0000 1060 0000 0000	OFCO 0000 0000 0000 0FD0 0000 0000 0000 0FE0 0000 0000 0000 0FF0 0000 0000 0000 1000 0001 0022 0005 1010 0037 0059 0008 1020 0000 0000 0000 1030 0000 0000 0000 1040 0000 0000 0000 1050 0000 0000 0000 1060 0000 0000 0000 1070 0000 0000 0000	OFCO 0000 0000 0000 0000 0000 OFDO 0000 0000 0000 0000 0000 OFEO 0000 0000 0000 0000 0000 OFFO 0000 0000 0000 0000 0000 1000 0001 0022 0005 0015 1010 0037 0059 0008 0000 1020 0000 0000 0000 0000 1030 0000 0000 0000 0000 1040 0000 0000 0000 0000 1050 0000 0000 0000 0000 1060 0000 0000 0000 0000 1070 0000 0000 0000 0000	OFCO 0000 0000 0000 0000 0000 0000 OFDO 0000 0000 0000 0000 0000 0000 0000 OFEO 0000 0000 0000 0000 0000 0000 0000 OFFO 0000 0000 0000 0000 0000 0000 0000 1000 0001 0022 0005 0015 0001 1010 0037 0059 0008 0000 0000 1020 0000 0000 0000 0000 0000 1030 0000 0000 0000 0000 0000 1040 0000 0000 0000 0000 0000 1050 0000 0000 0000 0000 0000 1060 0000 0000 0000 0000 0000 1070 0000 0000 0000 0000 0000	OFCO 0000 <th< th=""><th>OFCO 0000 <th< th=""><th>OFCO 0000 <th< th=""></th<></th></th<></th></th<>	OFCO 0000 <th< th=""><th>OFCO 0000 <th< th=""></th<></th></th<>	OFCO 0000 <th< th=""></th<>

Address	00	02	04	06	08	0A	0C	0E	ASCII
0FE0	0000	0000	0000	0000	0000	0000	0000	0000	
OFFO	0000	0000	0000	0000	0000	0000	0000	0000	
1000	0001	0005	0015	0001	0002	000D	0003	0022	".
1010	0037	0008	0059	0000	0000	0000	0000	0000	7Y
1020	0000	0000	0000	9000	0000	0000	0000	0000	
1030	0000	0000	9000	0000	0000	0000	0000	0000	
1040	0000	0000	0000	0000	0000	0000	0000	0000	
1050	0000	0000	0000	0000	0000	0000	0000	0000	
1060	0000	0000	0000	0000	0000	0000	0000	0000	
1000	0000	0000	000	0000	0000	0000	0000	0000	

After 1 bubble

Maximum Value Stred @ 0×1014

4. Develop a dspic executable program that will add two given (3x3) matrices and the result stores in a specified memory location.



```
1
       .equ __30F6010, 1
2
3
               .include "p30f6010.inc"
              .global __reset
4
5
6
           _reset:
8
9
           : MATRIX A
10
          MOV #1, W0
11
          MOV W0, 0x1000
12
          MOV #2, WO
          MOV W0, 0x1002
13
14
          MOV #3, W0
          MOV W0, 0x1004
          MOV #4, WO
16
          MOV W0, 0x1006
17
18
          MOV #5, W0
19
          MOV W0, 0x1008
          MOV #6, W0,
20
          MOV W0, 0x100A
21
22
          MOV #7, W0
          MOV W0, 0x100C
23
          MOV #8, W0
24
          MOV W0, 0x100E
25
26
          MOV #9, W0
27
          MOV W0, 0x1010
28
28
29
          : MATRIX B
30
          MOV #11, W0
31
          MOV W0, 0x1020
          MOV #12, W0
32
33
          MOV W0, 0x1022
34
          MOV #13, WO
          MOV W0, 0x1024
35
          MOV #14, W0
36
37
          MOV W0, 0x1026
38
          MOV #15, W0
39
          MOV W0, 0x1028
          MOV #16, W0,
40
41
          MOV W0, 0x102A
42
          MOV #17, WO
43
          MOV W0, 0x102C
          MOV #18, W0
44
45
          MOV W0, 0x102E
          MOV #19, W0
46
          MOV W0, 0x1030
47
48
                               ; PROGRAM COUNTER
49
          MOV #0x0000, W0
50
          MOV #0x1000, W1
                                  ; MATRIX A START ADDRESS
          MOV #0x1020, W2
                                  ; MATRIX B START ADDRESS
51
52
          MOV #0x1040, W3
                                 ; MATRIX C START ADDRESS
53
          LOOP1: DO #8, END1
                                 ; LOOP FOR 9 TIMES (SIZE OF MATRIX)
54
                                  ; TEMP STORE VALUE IN W4
55
          MOV [W1], W4
56
          MOV [W2], W5
                                  ; TEMP STORE VALUE IN W5
57
          ADD W4, W5, [W3]
                                 ; ADD W4, W5 STORE IN [W3]
58
          INC2 W1, W1
                                  ; INCREMENT ADDRESS OF W1
          INC2 W2, W2
                                  : INCREMENT ADDRESS OF W2
59
60
          INC2 W3, W3
                                 ; INCREMENT ADDRESS OF W3
61
          END1: INC WO, WO
                                  ; PROGRAM COUNTER INCREMENT
62
```

Matrix A \rightarrow shred from 0x1000 Matrix B \rightarrow shred from 0x1020 Matrix C = A+B \rightarrow shred from 0x1040

Before

701					DI CON	Pomes								
Address	00	02	04	06	08	0A	0C	0E	ASCII					
0FC0	0000	0000	0000	0000	0000	0000	0000	0000						
0FD0	0000	0000	0000	0000	0000	0000	0000	0000				۸ م ا	e	Λ
0FE0	0000	0000	0000	0000	0000	0000	0000	0000				Mar	XI/	1
OFF0	0000	0000	0000	0000	0000	0000	0000	0000		,,				
1000	0001	0002	0003	0004	0005	0006	0007	8000						
1010	0009	0000	0000	0000	0000	0000	0000	0000				. 4	1	_
1020	000B	000C	000D	000E	000F	0010	0011	0012			<i>, _</i>	Ma	kir	R
1030	0013	0000	0000	0000	0000	0000	0000	0000			//	11001		1
1040	0000	0000	0000	0000	0000	0000	0000	0000						
1050	0000	0000	0000	0000	0000	0000	0000	0000						
1060	0000	0000	0000	0000	0000	0000	0000	0000						
1070	0000	0000	0000	0000	0000	0000	0000	0000						
1080	0000	0000	0000	0000	0000	0000	0000	0000						
1090	0000	0000	0000	0000	0000	0000	0000	0000						
10A0	0000	0000	0000	0000	0000	0000	0000	0000						
			1	1	1			1						

Affer

Address	00	02	04	06	08	0A	0C	0E	ASCII
0FC0	0000	0000	0000	0000	0000	0000	0000	0000	
0FD0	0000	0000	0000	0000	0000	0000	0000	0000	
OFE0	0000	0000	0000	0000	0000	0000	0000	0000	
OFF0	0000	0000	0000	0000	0000	0000	0000	0000	
1000	0001	0002	0003	0004	0005	0006	0007	8000	
1010	0009	0000	0000	0000	0000	0000	0000	0000	
1020	000B	000C	000D	000E	000F	0010	0011	0012	
1030	0013	0000	0000	0000	0000	0000	0000	0000	
1040	000C	000E	0010	0012	0014	0016	0018	001A	
1050	001C	0000	0000	0000	0000	0000	0000	0000	
1060	0000	0000	0000	0000	0000	0000	0000	0000	
1070	0000	0000	0000	0000	0000	0000	0000	0000	
1080	0000	0000	0000	0000	0000	0000	0000	0000	
1090	0000	0000	0000	0000	0000	0000	0000	0000	

Makix

5. Develop a dspic executable program (i) that will generate a FIBONACCI series, (ii) average of the above series and stores in the memory.

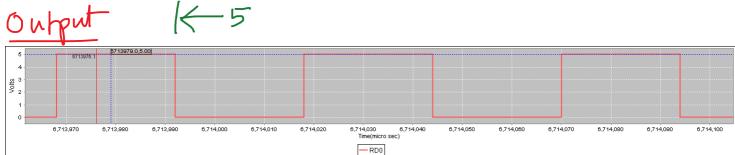
wde

```
.equ __30F6010, 1
2
            .include "p30f6010.inc"
            .global __reset
            .text
          reset:
         ; KEPT WO, W1 RESERVED BECAUSE DIVISION IS NEEDED
                        ; STARTING VALUE 0
         MOV #0x0000, W3
                            ; STARTING VALUE 1
10
         MOV #0x0001. W4
11
         MOV #0x1004, W2
                            ; MEMORY ADDRESS
13
         MOV W3, 0x1000
                            ; STORING 1ST VALUE
                            ; STORING 2ND VALUE
         MOV W4, 0x1002
14
15
16
         LOOP1: DO #12, END1
                          ; FIRST 15 FIBONACCI NUMBERS
         ADD W3, W4, W5
                            ; W3 = W1+W2
18
19
         MOV W5, [W2]
                            ; W1=0, W2=1, W3=1 AND SO ON
20
         INC2 W2, W2
                            ; INCREMENT ADDRESS
                            ; INCREMENT ADDRESS
21
         MOV W4, W3
22
         END1: MOV W5, W4
                            ; INCREMENT PROGRAM COUNTER
         : used till W5
23
24
         ; FOR AVERAGE CALCULATION
                            ; IMMEDIATE SUM
26
         MOV #0x1000, W6
         MOV #0x0000, W7
27
                            ; MIDDLE SUM
28
         MOV #0x0000, W8
                           ; FINAL SUM
29
         LOOP2: DO #14, END2
                            ; LOOP 15 TIMES
         ADD W7, [W6], W8
                            ; W8 = W7 + [W6]
30
         MOV W8, W7
END2: INC2 W6, W6
31
                            ; UPDATE VALUE
                            ; INCREMENT
32
         ; used till W8
33
34
35
         MOV #15, W9
                            : DENOMINATOR
36
         REPEAT #17
                            ; REQUIRED INSTRUCTION
37
         DIV.U W8, W9
                            ; DIVISON
          | |
           0000 0001 0001 0002 0003 0005 0008 000D .....
                                           0000 ..".7.Y. ....y...
                                      0000 0000 .....
 1030
           Address
          Symbol
                    Hex
                           Decimal Binary
                                                         Char
0000
          WREG0
                   0x0041 65
                                     000000000_010000001 '.A'
                                                                                 in Hex
                                     000000<del>00 90</del>001011 '...'
0002
          WREG1
                   0x000B 11
                                     00010000 00011110
0004
                   0x101E 4126
         WREG2
                                     00000000 11101001
0006
          WREG3
                   0x00E9 233
                                                                             Num = 65
                                     00000001 01111001 '.y'
0008
                   0x0179 377
         WREG4
                                     00000001 01111001
000A
          WREG5
                   0x0179 377
000C
                                     00010000 00011110 '..'
                   0x101E 4126
         WREG6
                                     00000011 11011010
000E
          WREG7
                   0x03DA 986
                                     00000011 11011010 '.Ú'
0010
         WREG8
                   0x03DA 986
0012
         WREG9
                   0x000F 15
                                     00000000 00001111 '..'
                                                                        =65.75 (dec)
                                     00000000 00000000 '...'
0014
          WREG10
                   0x0000 0
0016
                   0x0000 0
                                     00000000 000000000 '...'
         WREG11
0018
          WREG12
                   0x0000 0
                                     00000000 00000000 '...'
                  1^{6} 15 Fibonacu = 986 \Rightarrow Averger = 65.73
001A
          WREG13
```

Develop a dspic executable program to generate 100 kHz/50 kHz PWM signal using "I/O pins".

Code

```
#pragma config FPR = XTL
                                             // Primary Oscillator Mode (XTL)
      pragma config FOS = FRC
                                             // Oscillator Source (Internal Fast RC)
     *pragma config FCKSMEN = CSW_FSCM_OFF
                                            // Clock Switching and Monitor (Sw Disabled, Mon Disabled)
                                                                                                                  Configuration
Bits
     #pragma config FWPSB = WDTPSB 16
                                             // WDT Prescaler B (1:16)
     #pragma config FWPSA = WDTPSA_512
                                             // WDT Prescaler A (1:512)
     *pragma config WDT = WDT_OFF
                                             // Watchdog Timer (Disabled)
10
      // FBORPOR
11
     *pragma config FPWRT = PWRT_64
                                             // POR Timer Value (64ms)
13
     #pragma config BODENV = BORV20
                                             // Brown Out Voltage (Reserved)
     #pragma config BOREN = PBOR ON
                                             // PBOR Enable (Enabled)
14
     *pragma config LPOL = PWMxL ACT HIGH
15
                                             // Low-side FWM Output Polarity (Active High)
     #pragma config HPOL = PWMxH_ACT_HIGH
                                             // High-side FWM Output Polarity (Active High)
     *pragma config PWMPIN = RST IOPIN
                                             // PWM Output Pin Reset (Control with PORT/TRIS regs)
17
18
     #pragma config MCLRE = MCLR DIS
                                             // Master Clear Enable (Disabled)
19
     // FGS
20
21
     *pragma config GWRP = GWRP_OFF
                                             // General Code Segment Write Protect (Disabled)
22
     #pragma config GCP = CODE PROT OFF
                                             // General Segment Code Protection (Disabled)
23
24
      // FICD
     #pragma config ICS = ICS_PGD
                                            // Comm Channel Select (Use PGC/EMUC and PGD/EMUD)
25
26
27 🔁 // #pragma config statements should precede project file includes.
   // Use project enums instead of #define for ON and OFF.
30 = #include <xc.h>
31 | #include dibpic30.h>
32
33
     int main()
34 🗐 (
         // Configure all four port D pins (RDO, RD1, RD2, RD3)
36
         TRISD = 0b11111111111110000;
37
                                                               7 Ch 1 Duty
         // Set OC channel 1 pulse start and stop times
OCIR = 0;
39
     OC1R = 0;
40
41
42
                                                                  7 ch2 (not used here)
43
         // Set OC channel 2 pulse start and stop times
44
45
         OC2RS = 35;
46
          // Set output compare mode for continuous pulses
47
48
         OCICONbits.OCM = Oblo1;
         OC2CONbits.OCM = Ob101;
49
50
          // Configure timer 2 (default timer for output compare)
                                                                             50KH2 Freq/
         PR2 = 50; // 0.1ms period
T2CONbits.TON = 1; // Enable timer 2
52
53
         while(1)
55
56
             // endless loop
59
60
         return 0;
```



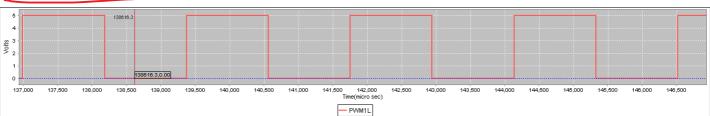


7. Develop a dspic executable program to generate 100 kHz/50 kHz PWM using PWM-channels of the processor.

```
17
           mov #0x0400, w0
                                   ; PWM module is disabled, continue operation in
19
           mov w0. PTCON
                                   ; Idle mode, special event interrupt disabled,
20
                                   ; immediate period updates enabled, no external
                                   ; synchronization
           ; Set the PWM Period
22
23
           mov #0x094D, w0
                                   ; Select period to be approximately 2.5?s
                                   ; PLL Frequency is ~480MHz. This equates to a
           mov w0, PTPER
25
                                   ; clocke period of 2.lns. The PWM period and
26
                                   ; duty cycle registers are triggered on both +ve
                                   ; and -ve edges of the PLL clock. Therefore,
28
                                   ; one count of the PTPER and PDCx registers
29
                                   ; equals 1.05ns.
                                   ; So, to achieve a PWM period of 2.5?s, we
31
                                   : choose PTPER = 0x094D
32
           ; Select individual Duty Cycle Control
                                  ; Fault interrupt disabled, Current Limit
           mov #0x0001, w0
           mov w0, PWMCON1
34
                                   ; interrupt disabled, trigger interrupt,
35
                                   ; disabled, Primary time base provides timing,
                                   ; DC1 provides duty cycle information, positive
37
                                   : dead time applied, no external PWM reset.
38
                                   ; Enable immediate duty cycle updates
           ; Duty Cycle Setting
           mov #0x094D, w0
                                   ; To achieve a duty cycle of 50%, we choose
40
41
           mov w0, PDC1
                                   ; the PDC1 value = 0.5*(PWM Period)
                                   ; The ON time for the PWM = 1.25?s
43
                                   ; The Duty Cycle Register will provide
44
                                   ; positive duty cycle to the PWMxH outputs
                                   ; when output polarities are active high
45
                                   ; (see IOCON1 register)
46
47
           bset PTCON, #15
                                   ; turn ON PWM module
```



4 look42 @ PWM1L



8. Develop a dspic executable program to generate 100 kHz PWM on the two PWM-channels of the processor with phase difference of (Ts/2, where f_s: 100 kHz).

wdl

```
equ __30F6010, 1
    3
                  .include "p30f6010.inc"
    4
                  .global __reset
    6
          ;Configuration bits:
    8
    10
                  config __FWDT, WDT_OFF
                                                     ;Turn off Watchdog Timer
    11
                  .text
                reset:
    13
    14
   15
              mov #0x0400, w0
                                      ; PWM module is disabled, continue operation in
   16
              mov w0, PTCON
                                      ; Idle mode, special event interrupt disabled,
    17
                                      ; immediate period updates enabled, no external
    18
                                      ; synchronization
              ; Set the PWM Period
    19
              mov #0x094D, w0
    20
                                      ; Select period to be approximately 2.5?s
              mov w0, PTPER
   21
                                      ; PLL Frequency is ~480MHz. This equates to a
   22
                                      ; clocke period of 2.lns. The PWM period and
   23
                                      ; duty cycle registers are triggered on both +ve
                                      ; and -ve edges of the PLL clock. Therefore,
    25
                                      ; one count of the PTPER and PDCx registers
   26
                                      ; equals 1.05ns.
   27
                                      ; So, to achieve a PWM period of 2.5?s, we
   28
                                      ; choose PTPER = 0x094D
    29
               ; Select individual Duty Cycle Control
    30
              mov #0xFFFF, w0
                                     ; Fault interrupt disabled, Current Limit
              mov w0, PWMCON1
    31
                                      ; interrupt disabled, trigger interrupt,
    32
                                      ; disabled, Primary time base provides timing,
   33
                                      ; DC1 provides duty cycle information, positive
    34
                                      ; dead time applied, no external PWM reset,
   35
                                      ; Enable immediate duty cycle updates
    36
    37
               ; Duty Cycle Setting
    38
               mov #0x094D, w0
                                      ; To achieve a duty cycle of 50%, we choose
               mov w0. PDC1
                                      : the PDC1 value = 0.5*(PWM Period)
    39
    40
                                      ; The ON time for the PWM = 1.25?s
    41
                                      ; The Duty Cycle Register will provide
                                      ; positive duty cycle to the PWMxH outputs
    43
                                      ; when output polarities are active high
    44
                                      ; (see IOCON1 register)
               mov #0x04A6, w0
                                      ; To achieve a duty cycle of 50%, we choose
    45
                                      ; the PDC1 value = 0.5*(PWM Period)
    46
               mov w0, PDC2
    47
                                      ; The ON time for the PWM = 1.25?s
    48
                                       ; The Duty Cycle Register will provide
                                      ; positive duty cycle to the PWMxH outputs
    50
                                      ; when output polarities are active high
                                      ; (see IOCON1 register)
    51
    52
           bset PTCON, #15 ; turn ON PWM module
    53
                                                       12 PWM @PWM1L & ZL
wave form
                                          42308.0
 Volts
```

 9. Develop a dspic executable program to generate two PWM signals on the "I/O pins" with finite time delay between them.

Code

\$10 × 2

/olts

```
*pragma config FPR = XTL
                                                               // Primary Oscillator Mode (XTL)
                 *pragma config FOS = FRC
                                                               // Oscillator Source (Internal Fast RC)
                 *pragma config FCKSMEN = CSW_FSCM_OFF // Clock Switching and Monitor (Sw Disabled, Mon Disabled)
                 #pragma config FWPSB = WDTPSB_16
#pragma config FWPSA = WDTPSA_512
                                                              // WDT Prescaler B (1:16)
                                                              // WDT Prescaler A (1:512)
                 *pragma config WDT = WDT_OFF
                                                              // Watchdog Timer (Disabled)
                 // FBORPOR
                 *pragma config FPWRT = PWRT_64
                                                              // POR Timer Value (64ms)
                 #pragma config BODENV = BORV20
#pragma config BOREN = PBOR_ON
           13
                                                              // Brown Out Voltage (Reserved)
                                                              // PBOR Enable (Enabled)
                 *pragma config LPOL = PWMxL ACT HIGH

*pragma config HPOL = FWMxH ACT HIGH

*pragma config PWMPIN = RST_IOPIN
                                                              // Low-side FWM Output Polarity (Active High)
                                                              // High-side PWM Output Polarity (Active High)
// PWM Output Pin Reset (Control with PORT/TRIS regs)
           18
                 #pragma config MCLRE = MCLR_DIS
                                                              // Master Clear Enable (Disabled)
           19
                 // FGS
           21
                 #pragma config GWRP = GWRP OFF
                                                              // General Code Segment Write Protect (Disabled)
                 *pragma config GCP = CODE_PROT_OFF
                                                              // General Segment Code Protection (Disabled)
           23
           24
           25
                 *pragma config ICS = ICS_PGD
                                                              // Comm Channel Select (Use PGC/EMUC and PGD/EMUD)
          27 U // *pragma config statements should precede project file includes.
28 U // Use project enums instead of *fdefine for ON and OFF.
          30 = #include <xc.h>
31 #include #include #include #include 
          32
          34 F (
                     // Configure all four port D pins (RDO, RD1, RD2, RD3)
          36
37
                     TRISD = 0b11111111111110000;
          39
                     // Set OC channel 1 pulse start and stop times
          41
                     OCIRS = 5;
          42
                     // Set OC channel 2 pulse start and stop times
          44
                     OC2R = 2:
                     OC2RS = 7;
                     // Set output compare mode for continuous pulses
          47
                     OCICONbits.OCM = 0b101;
          49
                     OC2CONbits.OCM = 0b101;
                     // Configure timer 2 (default timer for output compare) PR2\,=\,10\,; // 20us period
          52
                     T2CONbits.TON = 1; // Enable timer 2
          54
                     while (1)
          57
                         // endless loop
                                                  W Ps/2 phase Shift @ RDO
                     return 0;
Waveform
               942631.0 5.00
                942630
                         942640
                                                                                942700 942710
Instruction Cycles
                                                                                                   942720
                                                                                                           942730
                                                                                                                     942740
                                                                                                                              942750
                                                                                                                                       942760
                                                                                                                                                 942770
                                                                                                                                                          942780
```

- RD0 - RD1

10. Develop a dspic executable program to sense the external voltage signal using on-chip ADC and stores in the memory.

Cocle

```
#pragma config FWFSB = WDTFSB_16
#pragma config FWFSA = WDTFSA_512
#pragma config WDT = WDT_OFF
                                                                                                                                                                  // WDT Prescaler B (1:16)
// WDT Prescaler A (1:512)
// Watchdog Timer (Disabled)
  10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
                        #pragma config FPWRT = PWRT_64

#pragma config BODENV = BORV20

#pragma config BOREN = PBOR_ON
                                                                                                                                                                    // POR Timer Value (64ms)
// Brown Out Voltage (Reserved)
// PBOR Enable (Enabled)
                        **pragma config LPOL = PWNXL ACT_HIGH
**pragma config HPOL = PWNXH_ACT_HIGH
**pragma config FWMPIN = RST_IOPIN
**pragma config MCLRE = MCLR_DIS
                                                                                                                                                                    // Low-side PWM Output Polarity (Active High)
                                                                                                                                                                    // High-side PWR Output Polarity (Active High)
// PWR Output Pin Reset (Control with PORT/TRIS regs)
// Master Clear Enable (Disabled)
                        #pragma config GWRP = GWRP_OFF

#pragma config GCP = CODE_PROT_OFF
                                                                                                                                                                    // General Code Segment Write Protect (Disabled)
// General Segment Code Protection (Disabled)
                         *pragma config ICS = ICS_PGD
                                                                                                                                                                  // Comm Channel Select (Use PGC/EMUC and PGD/EMUD)
### Sinclude (xc.h)

### Sinclude (libpic30.h)

### sinclude (libpic30.h)

### unsigned int read_anal

### int main()

### // Declare a varia

### // Make RDD-3 digit

### // Configure analo

### //
                      unsigned int read_analog_channel(int n);
                                     // Declare a variable for the step time
// so that it can be changed easily
int v;
                                     long step_time = 300000L;
                                   // Make RD0-3 digital outputs
TRISD = 0b0000;
                                                                                                                                                                                                                                                                                                                                                                   Reads input @
POST B
                                    // Configure analog inputs

TRISB - OxOIFF; // Port B all inputs

ADPCFG - OxFF00; // Lowest 8 PORTB pins are analog inputs

ADCOMI = 0; // Manually clear SAMP to end sampling, start conversion

ADCOMI = 0; // Voltage reference from AVDD and AVS5

ADCOM3 = 0x0005; // Manual Sample, ADCS=5 -> Ted = 3* TeV = 0.lus

ADCOMIbits.ADON = 1; // Turn ADC ON
                                   // Cycle through the four \underline{\text{windings}} to make // the stepper turn forwards
                                                   // Read the analog channel. The result is an // integer between 0 and 1023 inclusive.
                                                   // integer between 0 and 10:
v = read_analog_channel(0);
   60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
                                                     // Now, update step time.
// Because the value get too big for 16-bit ints,
// the constant values are explicitly marked as
// long values so that the calculation is carried
// out using 32-bit ints.
step_time = 150000L + 200L * v;
                         // Cycle through the four stepper windings
LATD = 0b1000; delay32(step_time);
LATD = 0b0010; delay32(step_time);
LATD = 0b0010; delay32(step_time);
LATD = 0b0001; delay32(step_time);
    ADC Road
                                       ADCHS = channel;
                                     ADCHS = channel; // Select the requested channel
ADCONIDITE.SAMP = 1; // start sampling
_delay32(30); // lus delay @ 30 MIPS
ADCONIDITE.SAMP = 0; // start Converting
while (!ADCONIDITE.DONE); // Should take 12 * Tad = 1.2us
                                      return ADCBUFO;
```