Byeongchan Gwak, Student ID : 501026

Washington University in St. Louis

McKelvey School of Engineering

Fall Semester 2021

CSE467M: Embedded Computing Systems

Midterm Exam #2

1. **PIC16F882 system organization is shown below. On the chip block diagram shown below mark the following:**

   (a) Where the chip system software code image will be stored?
      a. EEPROM

   (b) Where the chip system clock and power will be connected?
      a. Clock/Power management

   (c) Where the chip system serial interface will be connected?
      a. Synchronous Serial Port

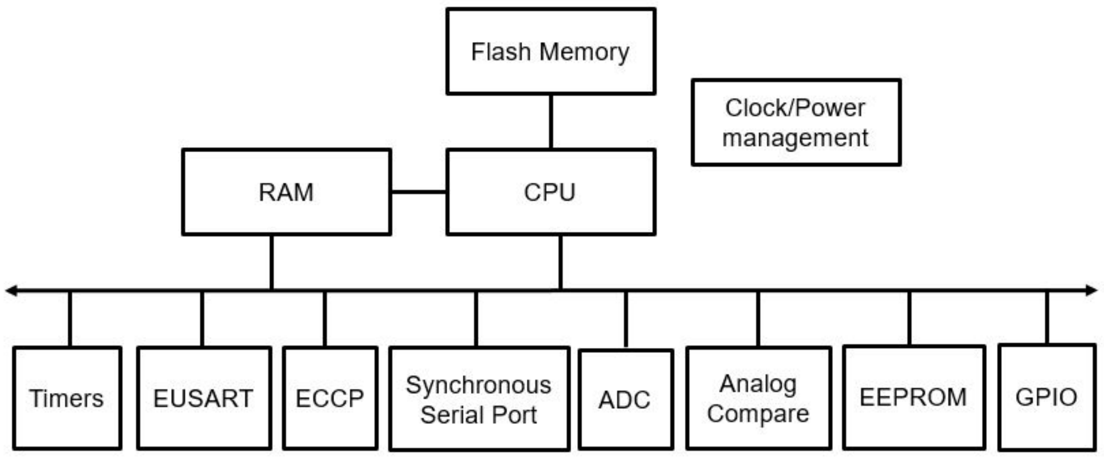
   (d) Where Analog-to-Digital Converter interface will be connected on the chip?
      a. Analogue to Digital Converter, or ADC

   (e) Where we can monitor some voltage level on the chip?
      a. General-purpose input/output (GPIO)

   (f) Where we can store some system important numbers used in the system?
      a. RAM

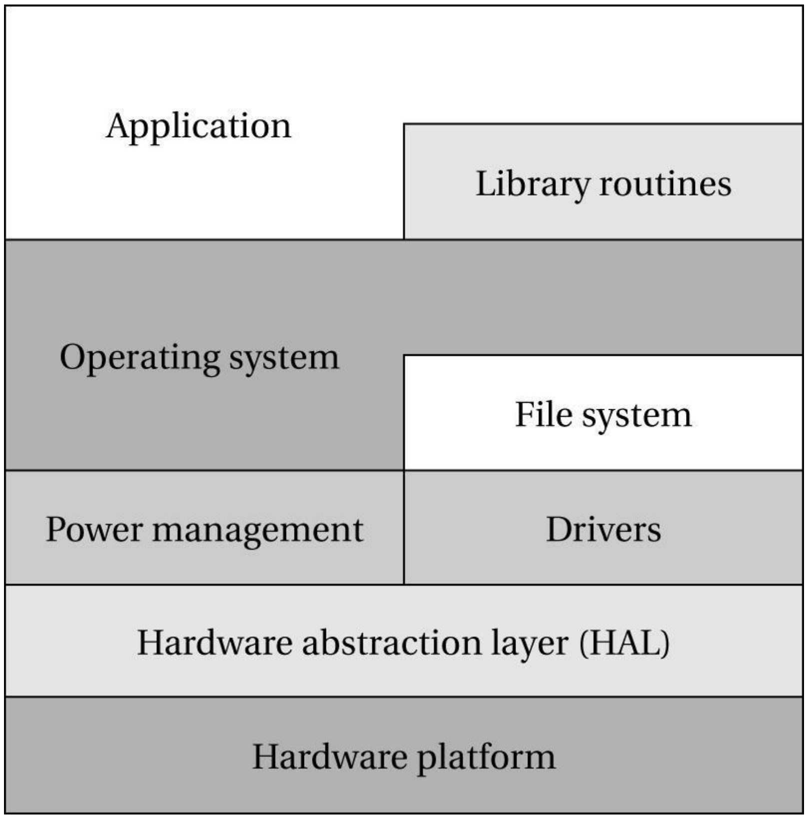
   Please be specific and explain your design.

   (30 points)



Byeongchan Gwak, Student ID : 501026

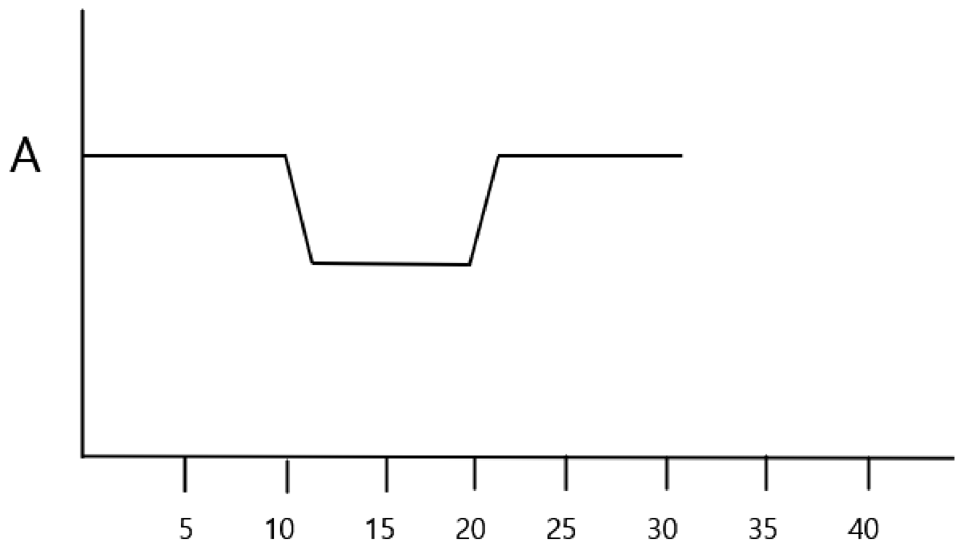
2. **Draw the software layer diagram for an embedded system and explain its all components from its hardware interface all the way up to system software apps. (20 points)**



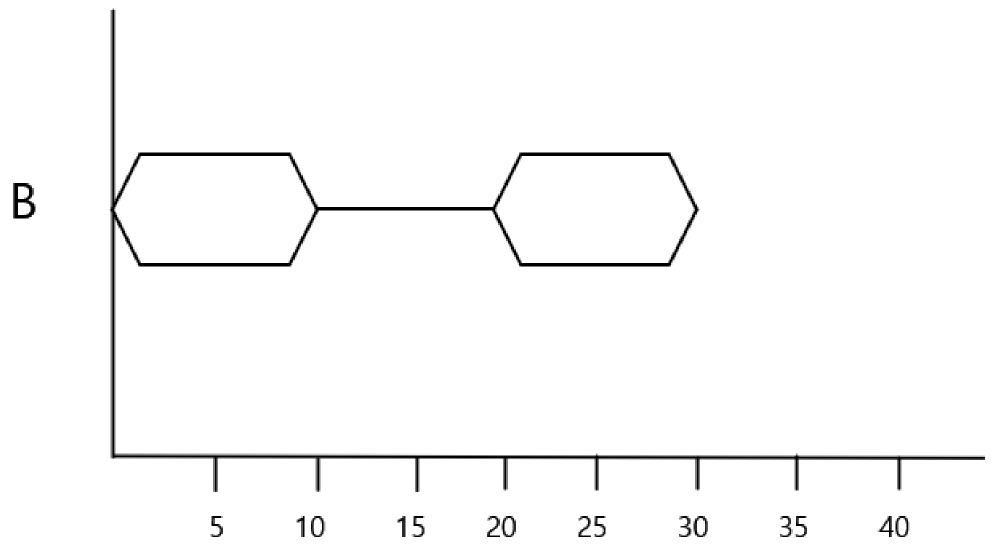
- HAL: Provides a basic level of abstraction from the hardware.
- Drivers: Use the HAL to simplify their structure and has low-level access to hardware.
- Power Management: Has low-level access to hardware
- Operating system & File system: Provide the basic abstractions required to build complex applications to make use of library routines to perform complex kernel functions.
- Application: Makes use of all these layers, either directly or indirectly.

Byeongchan Gwak, Student ID : 501026

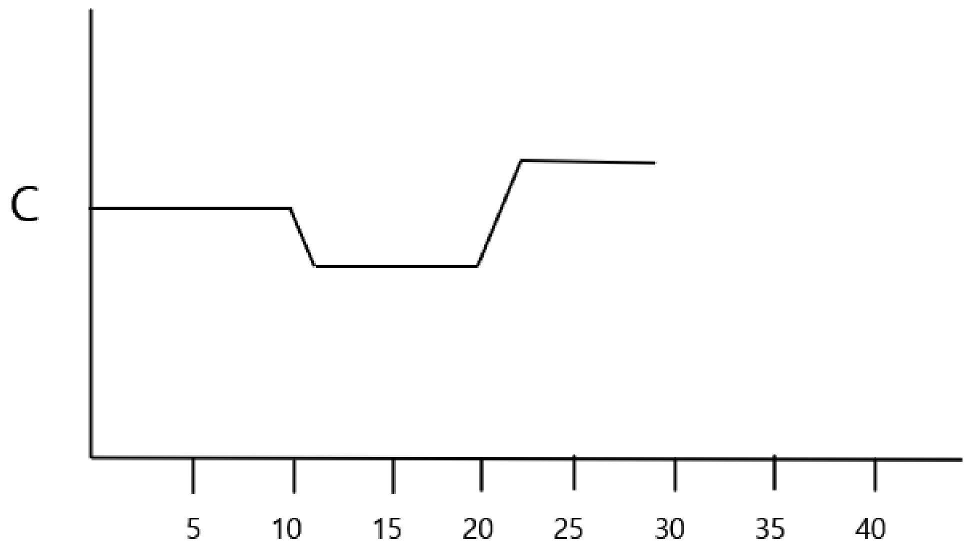
3. **Draw a timing diagram with the following signals (where [t_1, t_2] is the time interval starting at t_1 and ending at t_2):**
   (a) Signal A is 1 [0, 10], falling [10, 12], 0 [12, 20], rising [20, 22], 1 [22, 30]



   (b) Signal B is stable [0, 10], changing [10, 20], stable [20, 30]

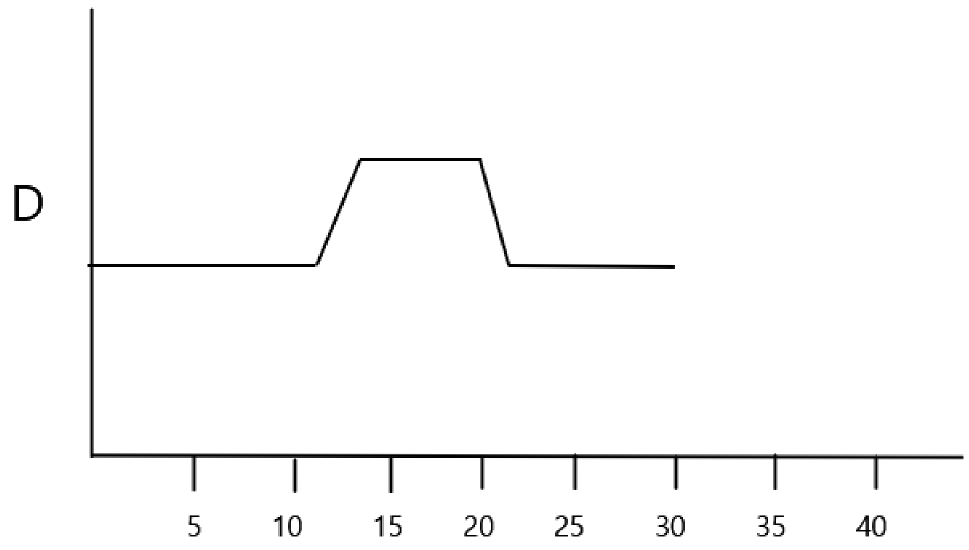


(c) Signal C is changing [0, 10], 0 [10, 20], rising [20, 22], 1 [22, 30]



(d) Signal D is 0 [0, 12], rising [12, 14], 1 [14, 20], falling [20, 22], 0 [22, 30]

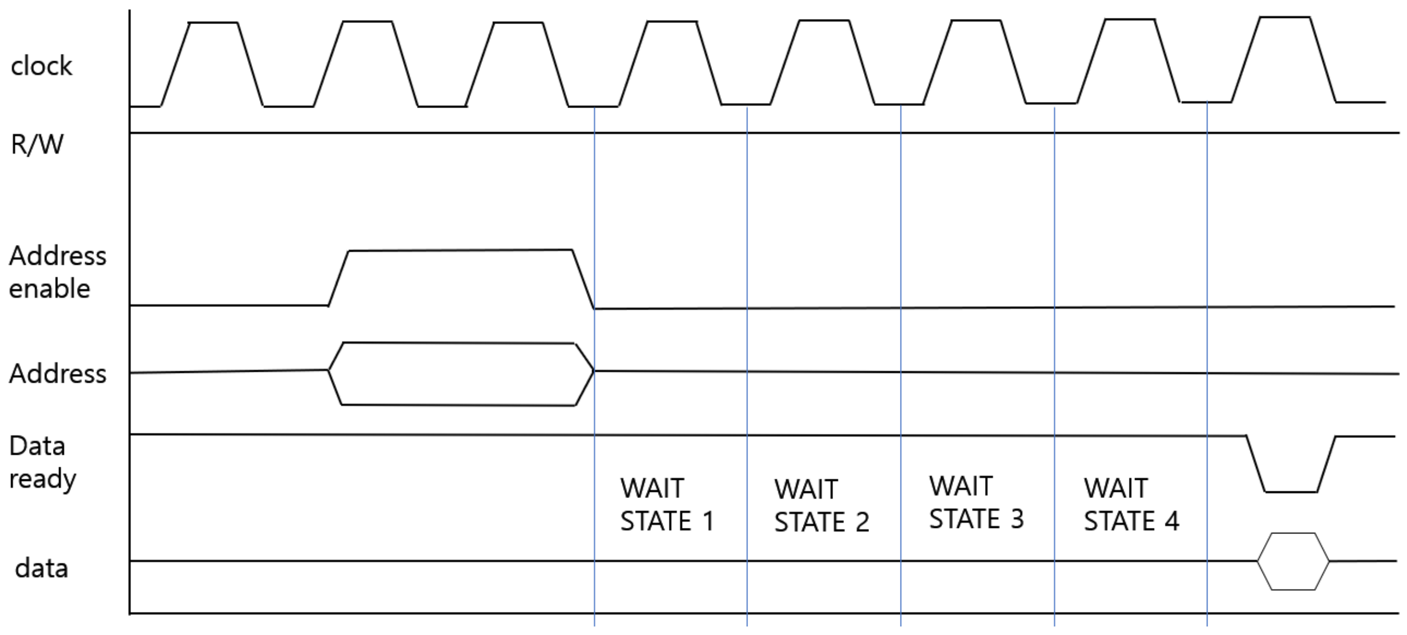
(40 points)



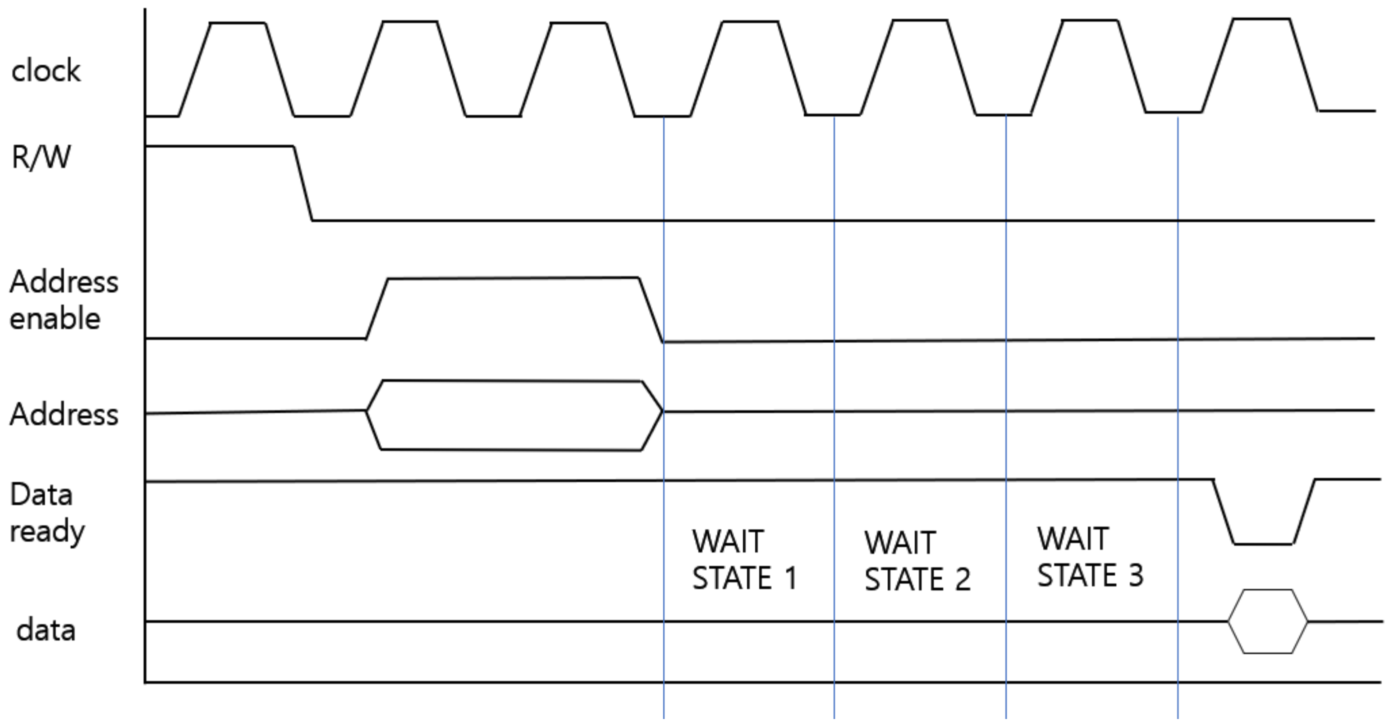
Byeongchan Gwak, Student ID : 501026

4. **Draw the following timing diagrams:**

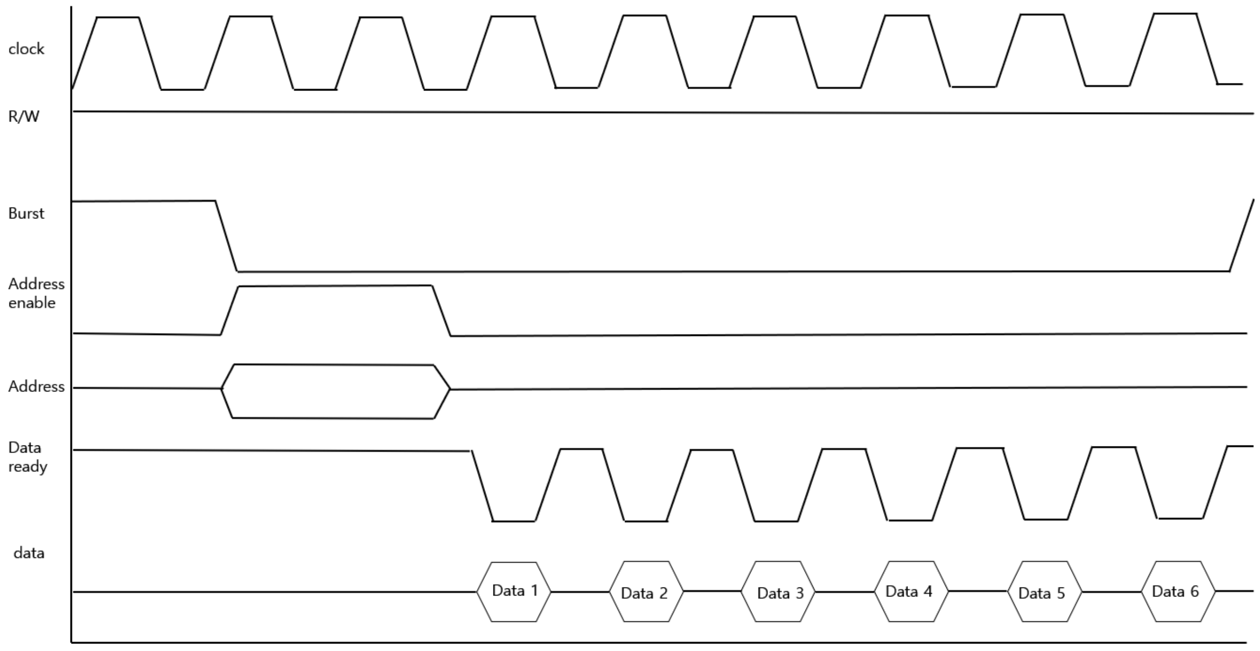
   (a) **Draw a timing diagram for a read operation with 4 wait states.**



   (b) **Draw a timing diagram for a write operation with 3 wait states.**



(c) Draw a timing diagram for a burst read operation that reads 6 locations. (30 points)



5. **Assume an A/D converter (ADC) is supplying samples at 20 kHz to an embedded system that has RISC processor with 10 MHz clock that executes 1 instruction per cycle.**

   (a) How much time is available per ADC sample for RISC processor instructions?

- 1/20kHz = 1/20,000Hz = 50 µs

   (b) If the RISC processor interrupt handler executes 150 instructions obtaining the sample and passing it onto the application routine and if the designed system code needs additional 250 instructions to process every sample from the A/D converter can this embedded system keep up real-time in processing A/D converter samples?

   (20 points)

- Total amount of time to deal with a sample

  10MHz = 100ns/instruction

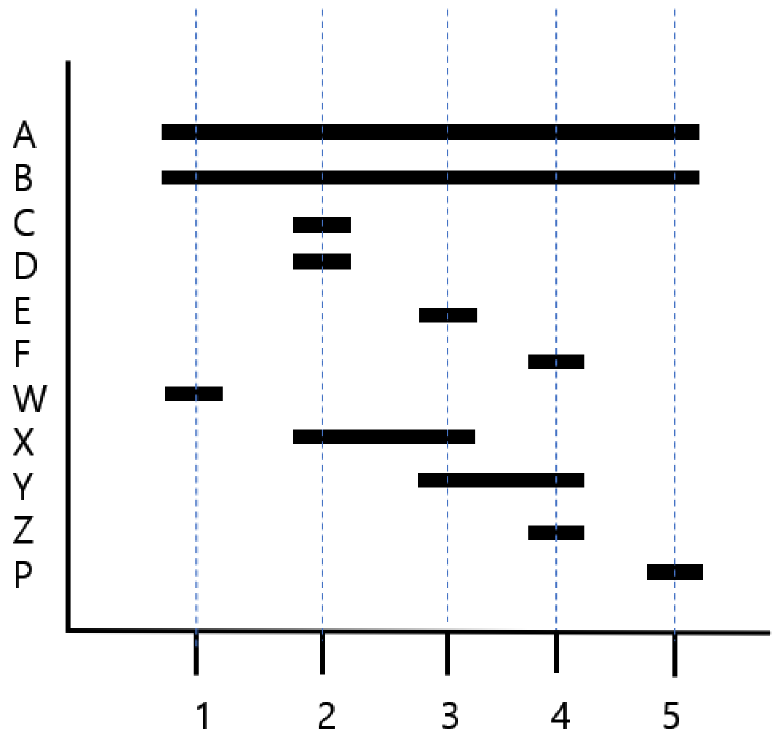
  (150 + 250) * 100ns = 40,000 ns

- ADC will supply samples at 50 µs = 50,000 ns
- So, ADC is supplying samples at 50,000ns and the RISC processor takes 40,000ns to deal with the samples.
- As a result, this embedded system can keep up real-time!!

6. Consider the C code shown below applied on some ARM's registers:

w = a + b; /* statement 1 */
x = c + d; /* statement 2 */
y = x + e; /* statement 3 */
z = y – f; /* statement 4 */
p = a – b; /* statement 5 */

(a) Draw lifetime register graph assuming that each variable from the C code gets one ARM's register. Determine what the optimum number of registers is needed for this C code from the lifetime register graph.

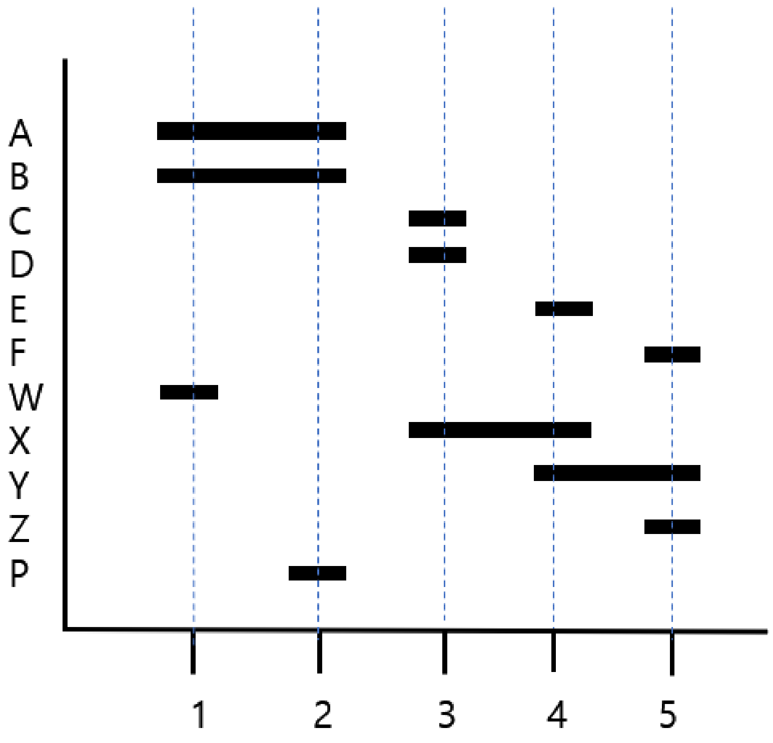


In this case we need 5 registers to implement. But we can reduce the number of registers by switching the statement order. If we move statement 5 after statement 1, and then the registers holding the A, B variables are no longer needed for the later.

(b) Now draw reduced lifetime register graph for the optimum number of registers. Also write modified C code for the optimum number of registers.

Place the statement 5 after statement 1. And then we can reduce the number of registers to use. We only need 3 registers to implement.

```
w = a + b; /* statement 1 */
p = a – b; /* statement 25 */
x = c + d; /* statement 32 */
y = x + e; /* statement 43 */
z = y – f; /* statement 54 */
```



Byeongchan Gwak, Student ID : 501026

(c) Write ARM's assembly language for each C code, original and modified and show the difference in usage of different number of registers.

(40 points)

In original version, we need to keep registers for a and b for later usage. Because statement 5 uses a and b variable again. Therefore, we need to use 5 registers. However, in modified version, don't need to keep them because the next statement inherits the value of a and b from previous statement. As a result, we only need 3 registers for the whole program.

Original version	Modified version

w = a + b; /* statement 1 */
x = c + d; /* statement 2 */
y = x + e; /* statement 3 */
z = y – f; /* statement 4 */
p = a – b; /* statement 5 */
	w = a + b; /* statement 1 */
	p = a – b; /* statement 25 */
	x = c + d; /* statement 32 */
	y = x + e; /* statement 43 */
	z = y – f; /* statement 54 */

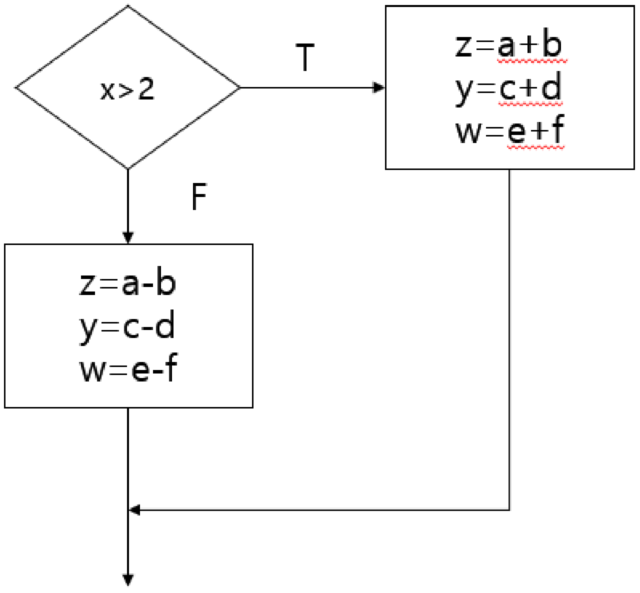
LDR r3, a LDR r4, b ADD r2, r3, r4 ; w = a + b STR r2, w ; store w LDR r0, c LDR r1, d ADD r2, r0, r1 ; x = c + d STR r2, x ; store x LDR r1, e ADD r0, r1, r2 ; y = x + e STR r0, y ; store y LDR r1, f SUB r2, r0, r1 ; z = y – f STR r2, z ; store z SUB r2, r3, r4 ; p = a – b STR r2, p ; store p	LDR r0, a LDR r1, b ADD r2, r0, r1 ; w = a + b STR r2, w ; store w SUB r2, r0, r1 ; p = a - b STR r2, p ; store p LDR r0, c LDR r1, d ADD r2, r0, r1 ; x = c + d STR r2, x ; store x LDR r1, e ADD r0, r1, r2 ; y = x + e STR r0, y ; store y LDR r1, f SUB r2, r0, r1 ; z = y – f STR r2, z ; store z

Byeongchan Gwak, Student ID : 501026

7. **Draw the CDFG for the following code fragments:**

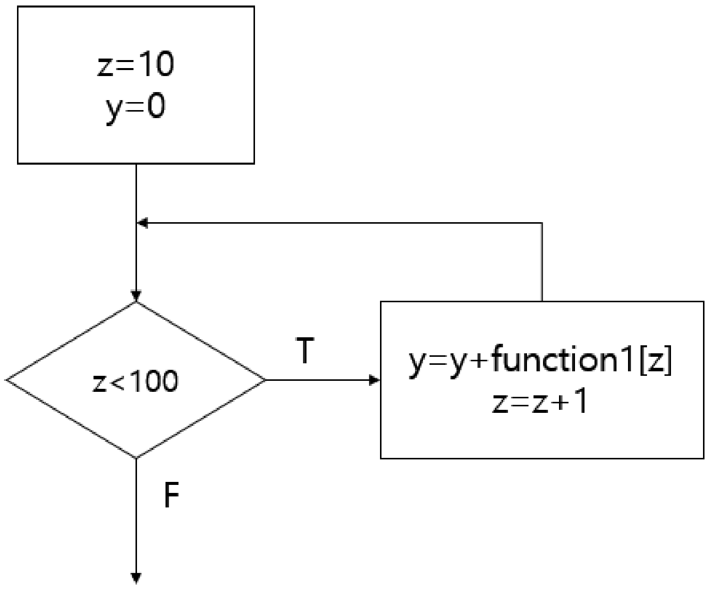
(a)

```
if (x > 2){
z = a + b;
y = c + d;
w = e + f;
}
else
{
z = a – b;
y = c – d;
w = e –f;
}
```



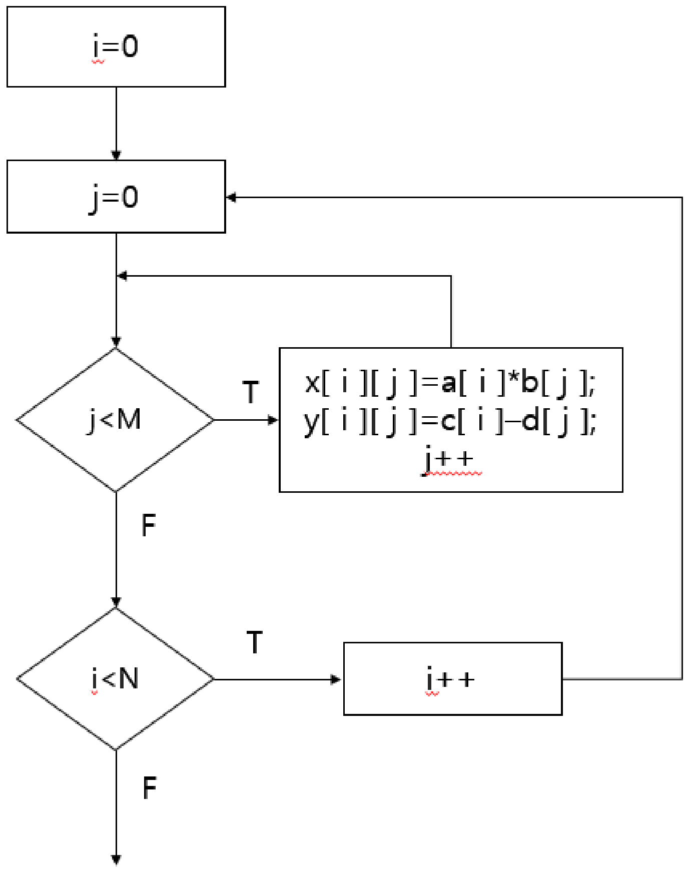
(b)

```
z = 10;
y = 0;
while( z < 100){
    y = y + function1[ z ];
    z = z + 1; }
```



(c)

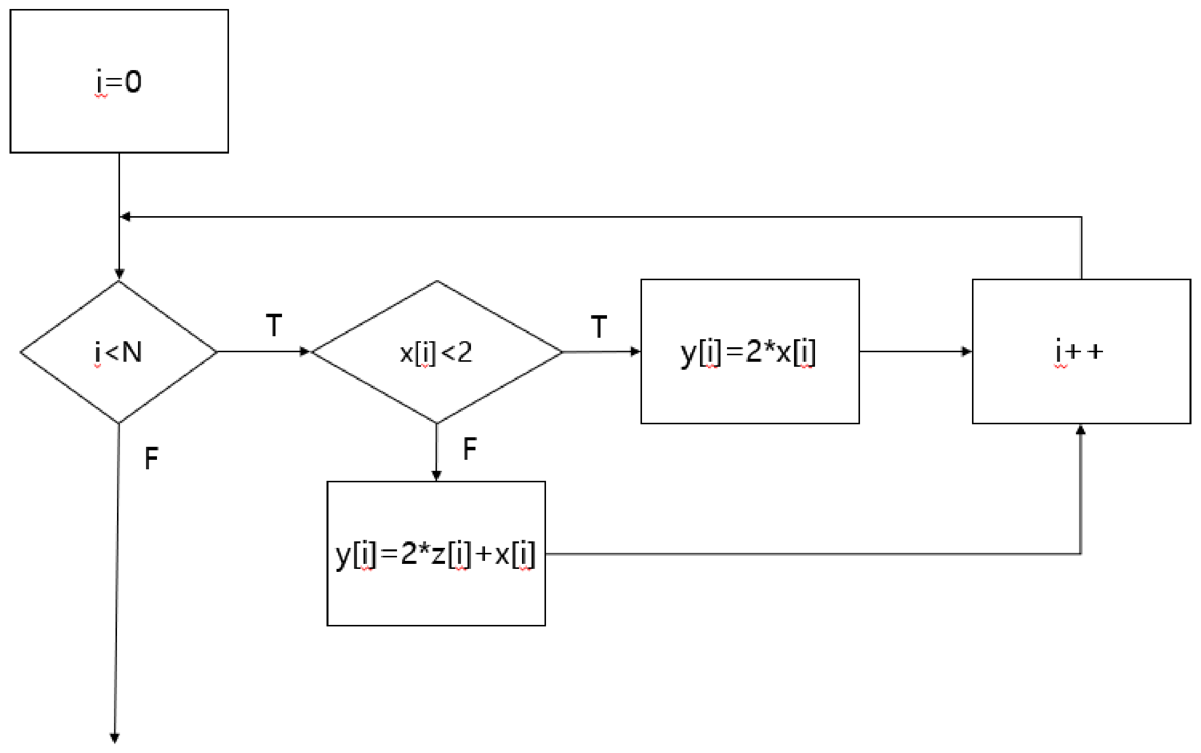
```
for (i = 0; i < N; i++){
    for (j = 0; j < M; j++){
        x[ i ][ j ] = a[ i ]*b[ j ];
        y[ i ][ j ] = c[ i ] – d[ j ];
    }
}
```



(d)

```
for (i = 0; i < N; i++){
      if ( x[ i ] < 2)
            y[ i ] = 2*x[ i ];
      else
           y[ i ] = 2*z[ i ] + x[ i ];
 }
```

(40 points)

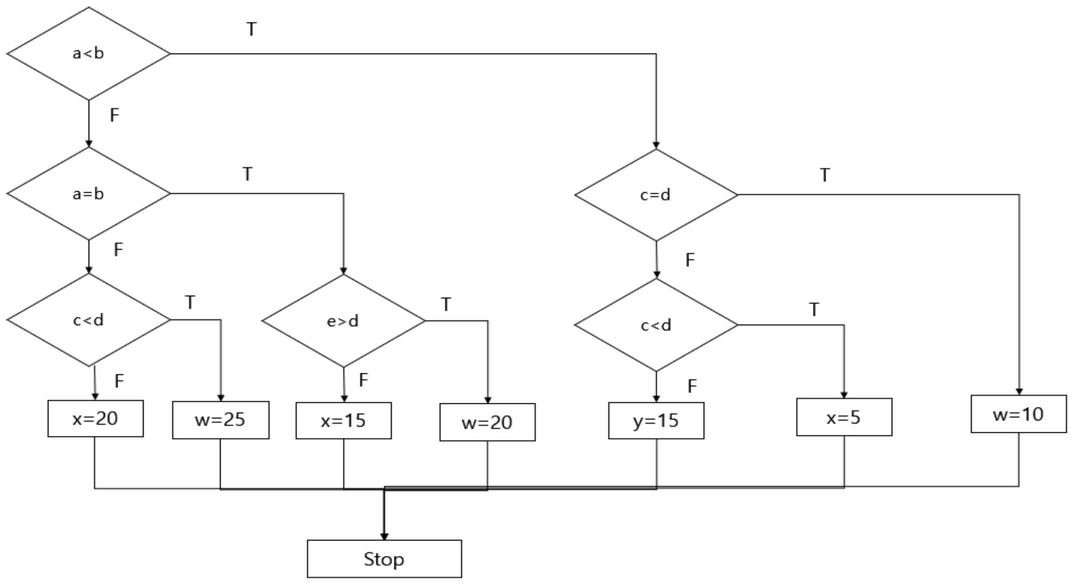


Byeongchan Gwak, Student ID : 501026

8. Find the cyclomatic complexity of the CDFGs for each of the code fragments shown below:

(a) C fragment code #1:

```
if ( a > b) {
        if (c = d)
            w = 10;
        elseif (c<d)
             x = 5;
        else
             y = 15; }
elseif (a = b) {
        if (e>d)
            w = 20
        else
             x = 15; }
else {
          if (c<d)
            w = 25;
          else
             x = 20;
     }
```



a.The cyclomatic complexity = e-n+2p

e: number of edges, n: number of nodes, p: number of components

e = 19, n = 14, p = 1

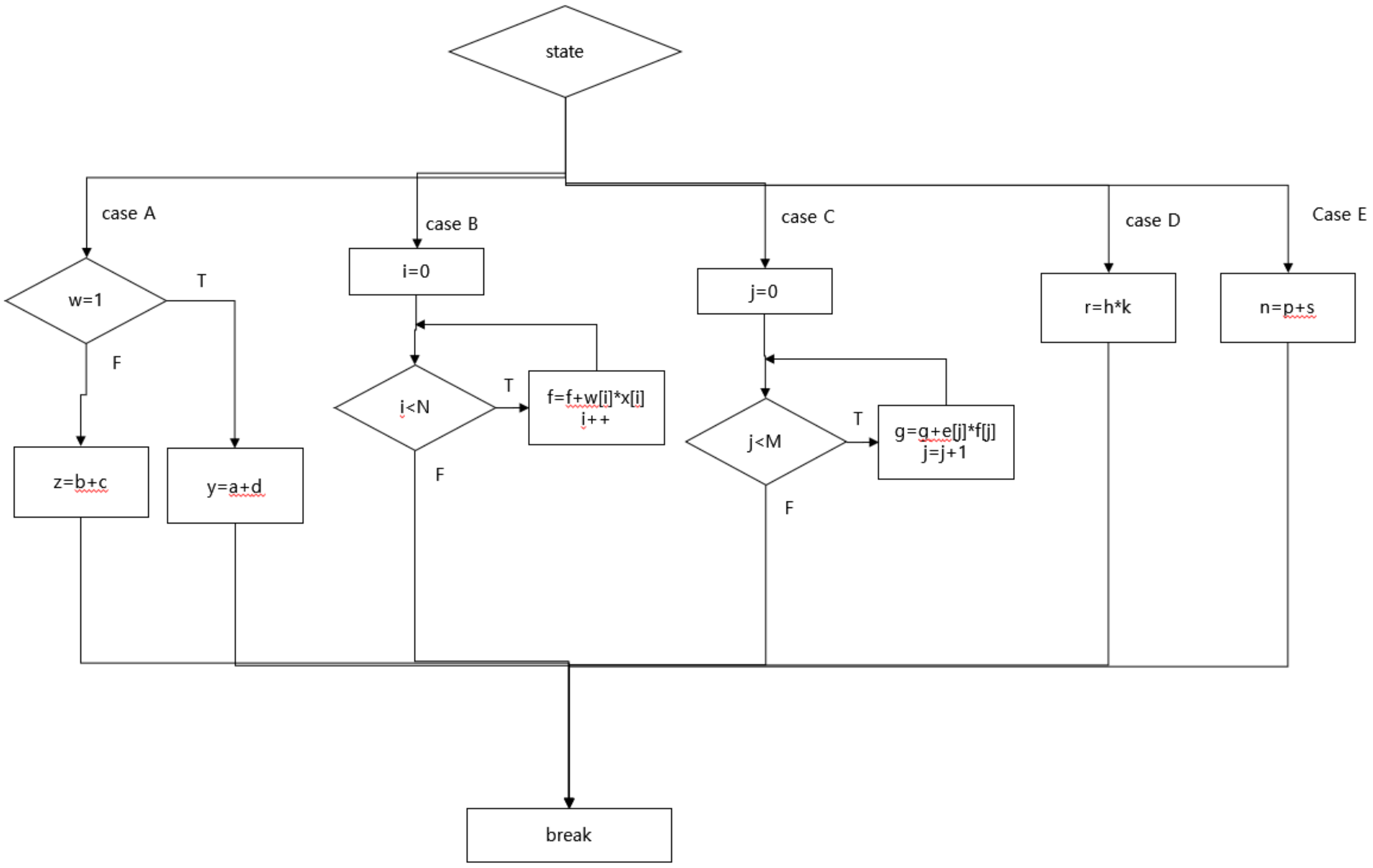
19 – 14 + 2 = 7

(b) C fragment code #2:

```
switch (state) {
        case A:
               if (w=1) { y = a+d; }
               else { z = b + c; }
               break;
        case B:
                for (i=0; i < N; i++)
                      f = f + w[i]*x[i];
                break;
        case C:
                j = 0;
                while (j<M) {
                         g = g + e[j]*f[j];
                         j = j +1; }
                 break;
        case D:
                 r = h*k;
                 break;
        case E:
                n = p+s;
        break; }
```

Please see the next page.

Byeongchan Gwak, Student ID : 501026



b.The cyclomatic complexity = e-n+2p

e: number of edges, n: number of nodes, p: number of components

e = 19, n = 13, p = 1

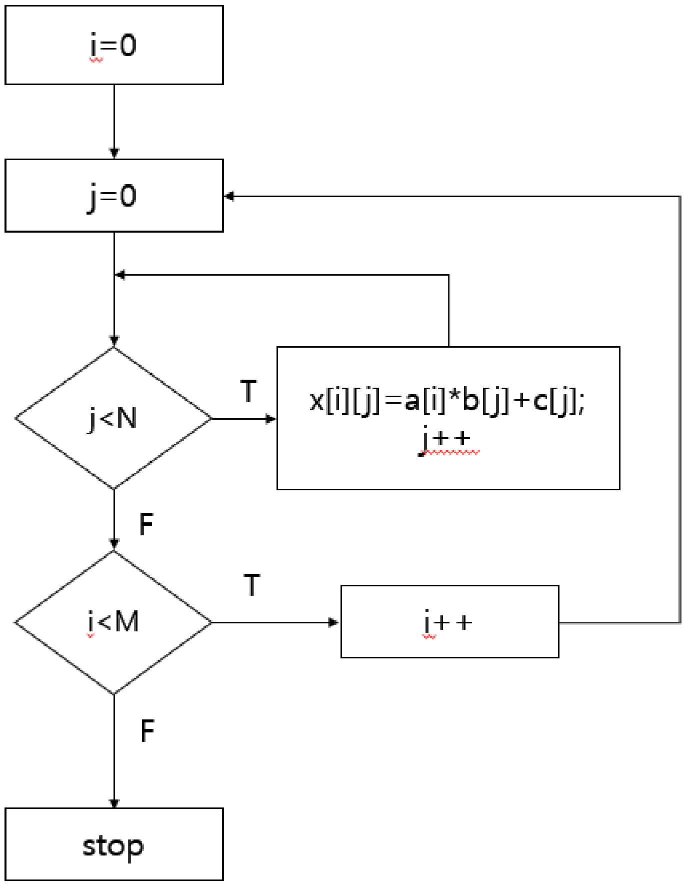
19 – 13 + 2 = 8

Byeongchan Gwak, Student ID : 501026

(c) C fragment code #3

```
for (i=0; i<M; i++)
     for (j=0; j<N; j++)
          x[i][j] = a[i]*b[j]+c[j];
```

(60 points)



c.The cyclomatic complexity = e-n+2p

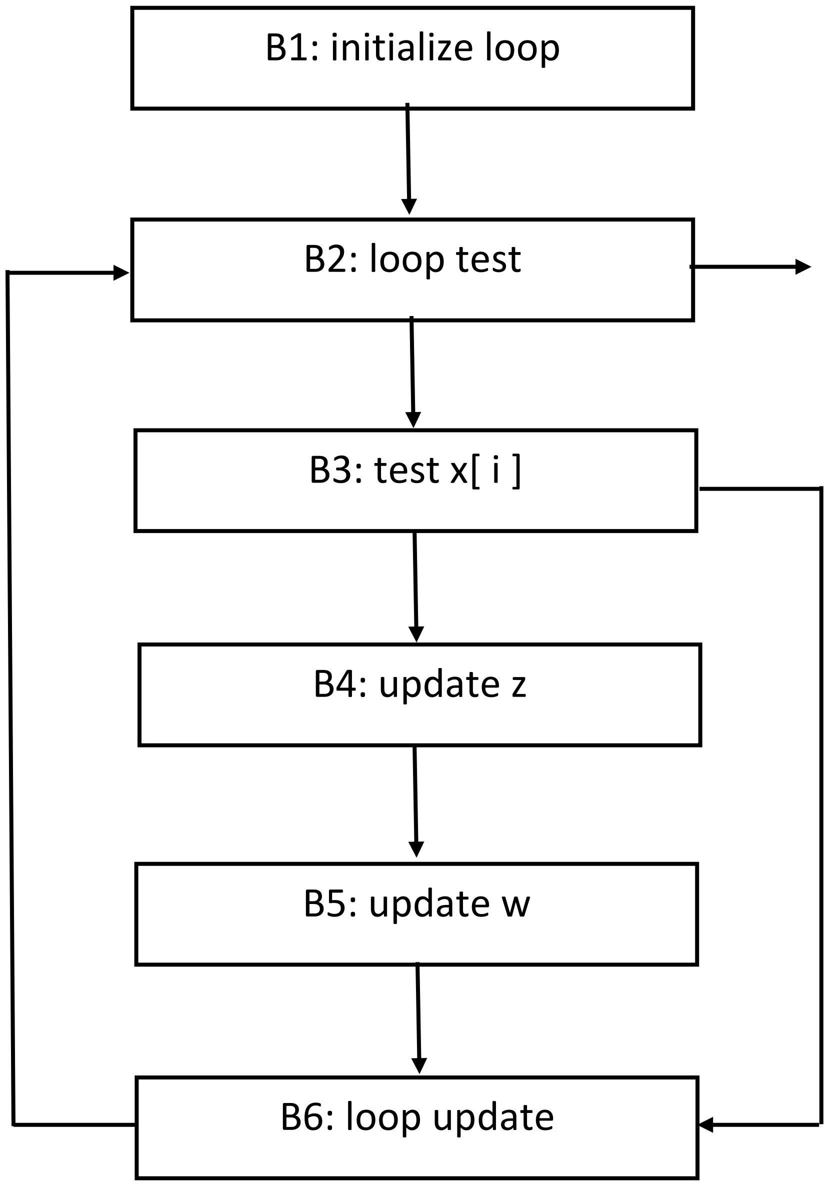
e: number of edges, n: number of nodes, p: number of components

e = 8, n = 7, p = 1

8 – 7 + 2 = 3

9. You are given this program and its flowchart.

```
for (i=0, z=0, w=0; i<25; i++){
      if (x[ i ] > 6){
              z = z + x[ i ]*y[ i ];
              w = w + x[ i ]+y[ i ];
              }
}
```



The execution time of the blocks is: B1 = 6 cycles, B2 = 2 cycles if branch taken, 5 cycles if not taken, B3 = 3 cycles if branch taken, 6 cycles if not taken, B4 = 7 cycles, B5 = 6 cycles and B6 = 1 cycle.

Byeongchan Gwak, Student ID : 501026

(a) What is the maximum number of times that each block in your flowchart executed?

Maximum number of times

B1 = 1 times (loop initialization)

B2 = 25 times (variable i from 0 to 24, the total is 25)

B3 = 25 times (if case is also will be tested 25 time as well)

B4 = 25 times (maximum is when if condition is true)

B5 = 25 times (maximum is when if condition is true)

B6 = 25 times (after for loop, increment i by one)

(b) What is the minimum number of times that each block in your flowchart executed?

Minimum number of times

B1 = 1 times (loop initialization)

B2 = 25 times (variable i from 0 to 24, the total is 25)

B3 = 25 times (if case is also will be tested 25 time as well)

B4 = 0 times (minimum is when if condition is false)

B5 = 0 times (minimum is when if condition is false)

B6 = 25 times (after for loop, increment i by one)

(c) What is the maximum execution time of the program in the clock cycles?

B1 = 1 times * 6 cycles = 6

B2 = 25*2 + 5(when not taken) = 55

B3 = 25*3 = 75

B4 = 25*7 = 175

B5 = 25*6 = 150

B6 = 25 * 1 = 25

The total = 6 + 55 + 75 + 175 + 150 + 25 = 486 cycles

(d) What is the minimum execution time of the program in the clock cycles? (60 points)

B1 = 1 times * 6 cycles = 6

B2 = 25*2 + 5(when not taken) = 55

B3 = 25*6 = 150

B4 = 0*7 = 0

B5 = 0*6 = 0

B6 = 25 * 1 = 25

The total = 6 + 55 + 150 + 0 + 0 + 25 = 236 cycles

Byeongchan Gwak, Student ID : 501026

10. What are 6 software optimization tasks that can be used for improvement of energy consumption of embedded system. Please describe all of them.

(30 points)

1. Use register efficiently. For example, group accesses to a value together so as to bring values together into register.
2. Resolve major cache conflict.
   A. Instruction conflict: Try to rewrite the segment to fit into the cache.
   B. Scalar data conflict: Move the data values to different locations to avoid conflicts
   C. Array data conflict: Moving the arrays or changing array access patterns to reduce conflicts.
3. Use page mode access in memory whenever possible. Page mode can reduce one step in memory access, saving a considerable amount of power.
4. Loop unrolling. However, too much unrolling could increase power consumption due to the lower hit rates of straight-line code.
5. Use software pipeline stalls to reduce the average energy per instruction.
6. Try to avoid recursive procedure calls. Function call overhead consumes much power. For example, tail recursion can often be eliminated.

Byeongchan Gwak, Student ID : 501026

11. Compute the utilization for these task sets:

CPU Utilization = (CPU time for useful work) / (Total available CPU time)

And it is important to unify the units.

(a) P1: period = 100 ms, execution time = 10 ms;
P2: period = 200 ms, execution time = 50 ms;
P3: period = 400 ms, execution time = 100 ms;

10/100 + 50/200 + 100/400 = 0.1 + 0.25 + 0.25 = 0.6

(b) P1: period = 1 s, execution time = 10 ms;
P2: period = 200 ms, execution time = 80 ms;
P3: period = 500 ms, execution time = 100 ms;
P4: period = 250 ms, execution time = 25 ms;

10/1000 + 80/200 + 100/500 + 25/250 = 0.71

(c) P1: period = 1 s, execution time = 100 ms;
P2: period = 2 s, execution time = 200 ms;
P3: period = 4 s, execution time = 400 ms;
P4: period = 5 s, execution time = 500 ms;
P5: period = 10 s, execution time = 1 s;

100/1000 + 200/2000 + 400/4000 + 500/5000 + 1/10 = 0.5

(d) P1: period = 1 μs, execution time = 10 ns;
P2: period = 2 μs, execution time = 50 ns;
P3: period = 4 μs, execution time = 100 ns;
P4: period = 5 μs, execution time = 500 ns;
P5: period = 10 μs, execution time = 100 ns;
P6: period = 20 μs, execution time = 400 ns;

(20 points)

10/1000 + 50/2000 + 100/4000 + 500/5000 + 100/10000 + 400/20000 = 0.19

Byeongchan Gwak, Student ID : 501026

12. Apply Rate-Monotonic Schedule (RMS) Scheduling on the following set of processes:

(a)

Process	**Execution time**	**Period**
P1	**1**	**4**
P2	**1**	**6**
P3	**2**	**8**
P4	**3**	**12**
P5	**2**	**24**

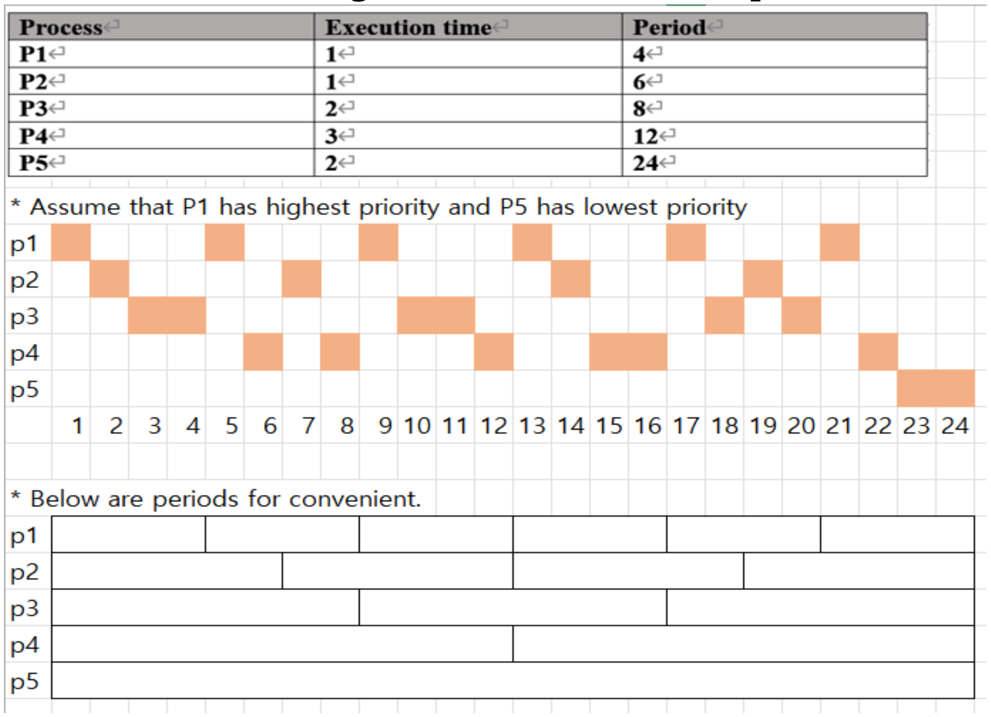
What is the least-common multiplier of the periods?

LCM = 24

Is this feasible scheduling example? => YES

Process	Execution time	Period	Executed count (LCM=24)	Units of CPU time
P1	1	4	6	1*6 = 6
P2	1	6	4	1*4 = 4
P3	2	8	3	2*3 = 6
P4	3	12	2	3*2 = 6
P5	2	24	1	2*1 = 2
			Total units of CPU time	6 + 4 + 6 + 6 + 2 <= 24 It is feasible! Because it doesn't exceed the total CPU time.

Draw the timeline diagram with the scheduled processes.



Process	**Execution time**	**Period**
P1	**1**	**4**
P2	**1**	**6**
P3	**2**	**8**
P4	**3**	**12**
P5	**2**	**24**

* Assume that P1 has highest priority and P5 has lowest priority

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
p1	🟧				🟧				🟧				🟧				🟧				🟧			
p2		🟧					🟧							🟧					🟧					
p3			🟧	🟧						🟧	🟧							🟧		🟧				
p4						🟧		🟧				🟧			🟧	🟧						🟧		
p5																							🟧	🟧

* Below are periods for convenient.

(b)

Process	**Execution time**	**Period**
P1	**2**	**4**
P2	**1**	**6**
P3	**2**	**8**
P4	**3**	**12**

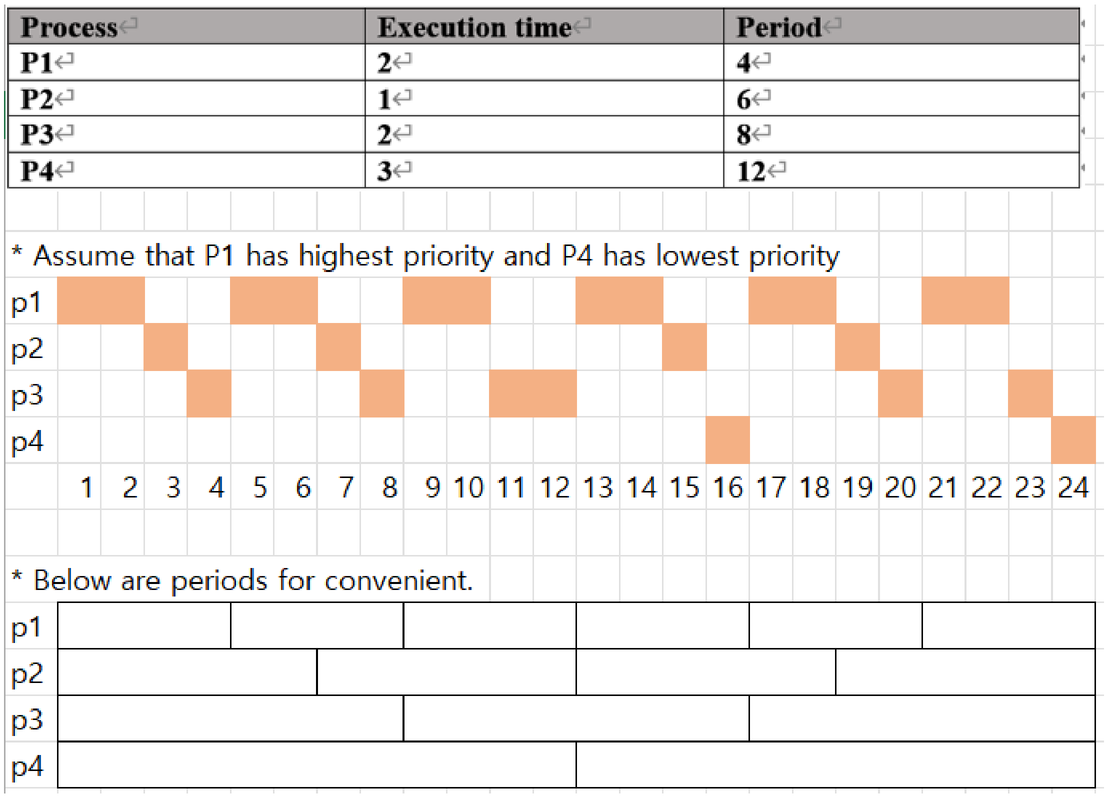
What is the least-common multiplier of the periods?

LCM = 24

Is this feasible scheduling example? => NO

Process	Execution time	Period	Executed count (LCM=24)	Units of CPU time
P1	2	4	6	2*6 = 12
P2	1	6	4	1*4 = 4
P3	2	8	3	2*3 = 6
P4	3	12	2	3*2 = 6
			Total units of CPU time	12 + 4 + 6 + 6 > 24 It is NOT feasible! Because it exceeds the total CPU time.

Draw the timeline diagram with the scheduled processes.
(60 points)



Process	**Execution time**	**Period**
P1	**2**	**4**
P2	**1**	**6**
P3	**2**	**8**
P4	**3**	**12**

* Assume that P1 has highest priority and P4 has lowest priority

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
p1	▨	▨			▨	▨			▨	▨			▨	▨			▨	▨			▨	▨		
p2			▨				▨								▨				▨					
p3				▨				▨			▨	▨								▨			▨	
p4																▨								▨

* Below are periods for convenient.

13. Apply Earliest-Deadline-First (EDF) Scheduling on the following set of processes:

Process	Execution time	Period
P1	**1**	**4**
P2	**2**	**5**
P3	**2**	**8**

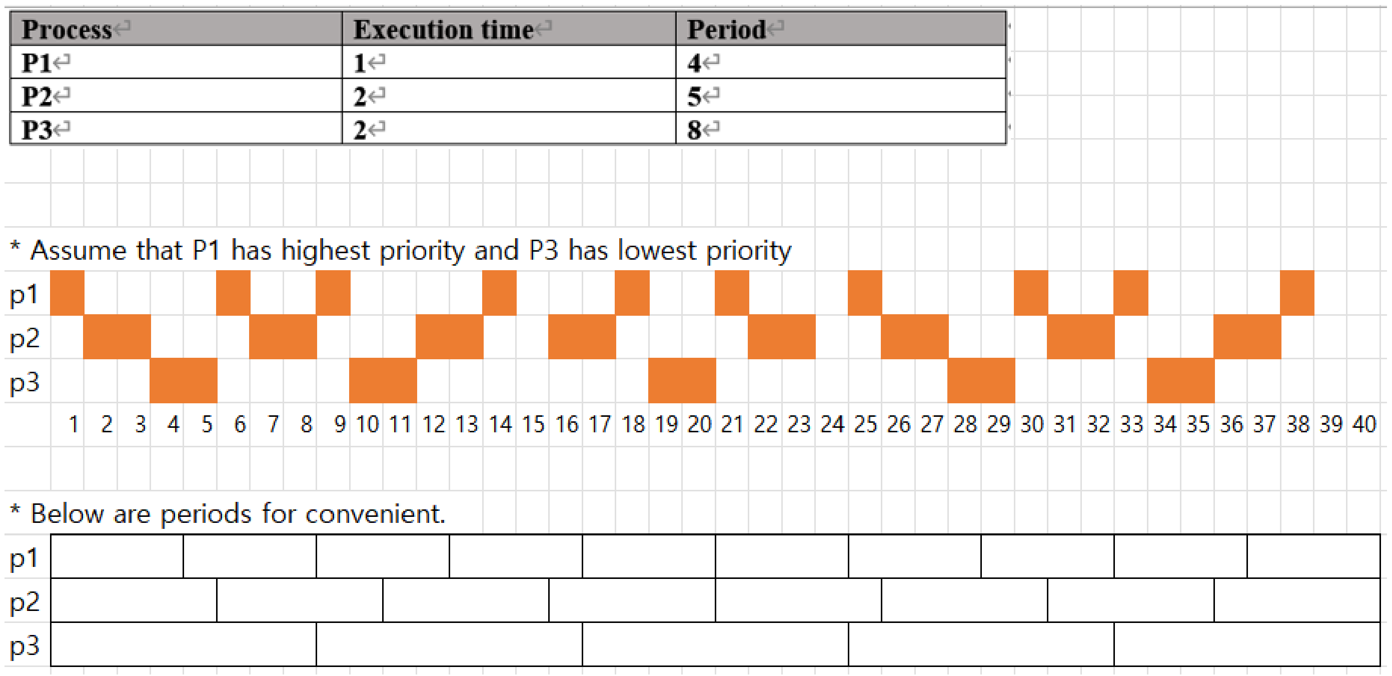
What is the least-common multiplier of the periods?

LCM = 40

Is this feasible scheduling example? => **YES**

Process	Execution time	Period	Executed count (LCM=40)	Units of CPU time
P1	1	4	10	1*10 = 10
P2	2	5	8	2*8 = 16
P3	2	8	5	2*5 = 10
			Total units of CPU time	10 + 16 + 10> 50 It is feasible! Because it doesn't exceeds the total CPU time.

Draw the timeline table using EDF Scheduling.
(30 points)

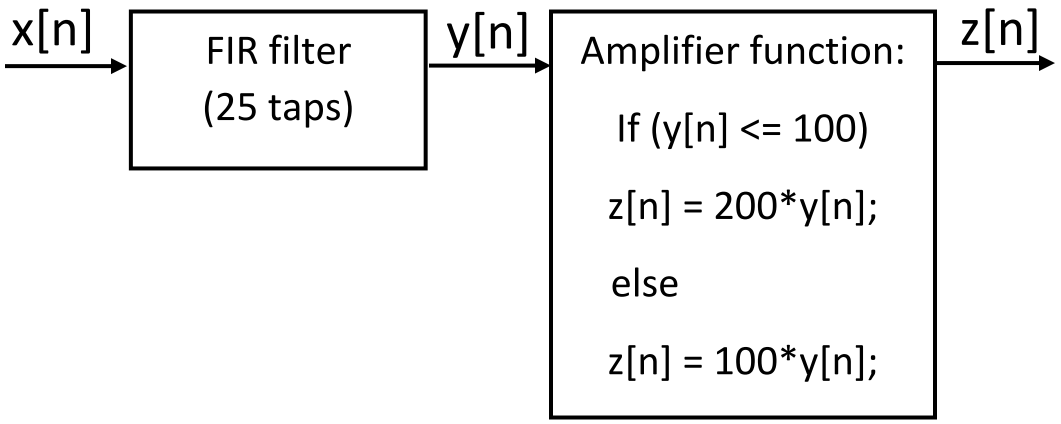


Process↵	**Execution time**↵	**Period**↵
P1↵	**1**↵	**4**↵
P2↵	**2**↵	**5**↵
P3↵	**2**↵	**8**↵

* Assume that P1 has highest priority and P3 has lowest priority

* Below are periods for convenient.

14. Using Programming Example 5.3 – An FIR Filter in C, from the textbook on pages 227 and 228 do the following software design:



(1) Design FIR filter in C that will use 24 previous input samples plus current input sample to calculate current output. Draw FIR filter diagram.

(2) FIR filter will have 25 coefficients b[0] thru b[24].

(3) You will need to use circular buffer for this design just like it is done in Example 5.3.

(4) Please comment all your software development.

(150 points)

- **FIR filter diagram**

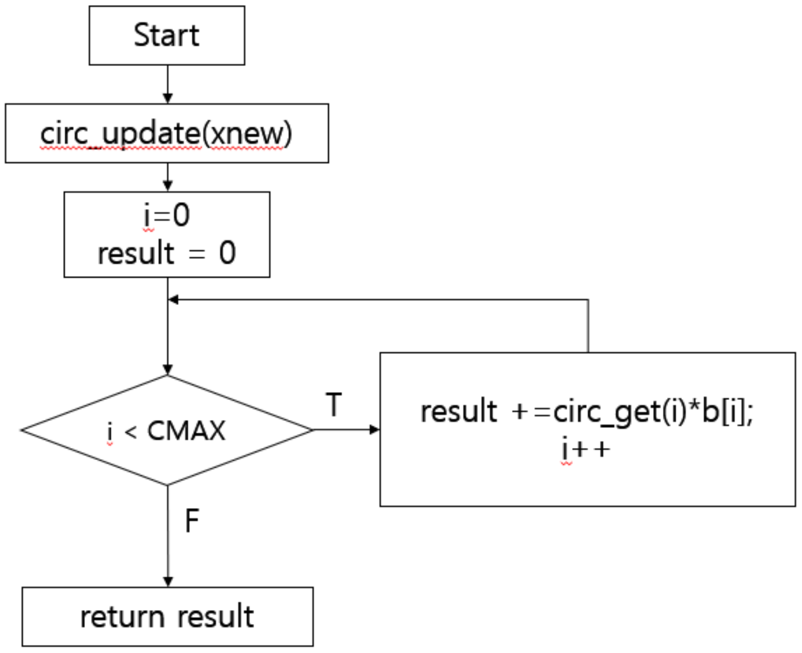


Diagram for FIR filter

Byeongchan Gwak, Student ID : 501026

- Explanation for all of my software
  - Main() : Reads input sample values from INPUT array and calls fir() & amplifier() functions.
  - fir() : This is a FIR filter funtion. It adds a sample value ‘xnew’ into circular array and calculate FIR values by multiply coefficient values(b array).
  - amplifier(): This is a literally amplifier. It amplifies the value according to the input value.
  - circ_update() : Inserting a new value and then push off the head position by one.
  - circ_get() : Access circular array to get a sample value descending order because a most recent sample is at the top the circular array.

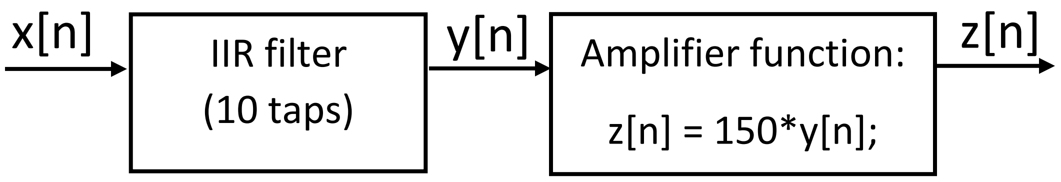
- Below is FIR implementation

```c
#define INPUT_LEN   50 /* input sample length */
#define CMAX        25 /* filter order */

int b[CMAX];    /* coefficient */
int x[CMAX];    /* circular buffer for sample */
int y[CMAX];    /* output for after fir */
int z[CMAX];    /* output for after amplifier */
int pos;        /* position of current sample */

/* main function for fir & amplifier */
void main() {
    circ_init();
    int n;
    /* call fir() & amplifier() for each input sample value */
    for(n=0; i<INPUT_LEN; n++) {
        y[n] = fir(INPUT[n]);
        z[n] = amplifier(y[n]);
    }
}

/* initialize buffer */
void circ_init() {
    int i;
    for(i=0;i<CMAX;i++) {
        x[i] = 0;
        y[i] = 0;
        z[i] = 0;
    }
    pos = CMAX-1;
}
```

Byeongchan Gwak, Student ID : 501026

```c
/* given a new sample value, update the buffer and compute the FIR output */
int fir(int xnew) {
    int i, result;

    circ_update(xnew);
    for(i=0, result=0; i<CMAX; i++) {
        /* Calculate FIR value */
        /* input value * coefficient b value */
        result += circ_get(i) * b[i];
    }

    return result;
}

/* amplify the result of FIR filter */
int amplifier(int fir) {
    int result = 0;
    if(fir <= 100) {
        result = 200 * y[i];
    }
    else {
        result = 100 * y[i];
    }
    return result;
}

/* add a new value and push off the oldest one */
void circ_update(int xnew) {
    /* the position moves from 0 to CMAX-1 */
    pos = ((pos == CMAX-1) ? 0 : (pos+1));
    x[pos] = xnew;
}

/* get the ith value from the buffer */
int circ_get(int i) {
    int ii = (pos-i) % CMAX;
    /* when result of modulo goes down minus
       add CMAX to make circular buffer */
    if(ii < 0) ii += CMAX;
    return x[ii];
}
```

15. Using Programming Example 5.4 – A Direct Form II IIR Filter Class in C, from the textbook on pages 228 and 229 do the following software design:



(1) Design IIR filter in C that will use 9 previous input samples plus current input sample to calculate current output. Draw IIR filter diagram.
(2) IIR filter will have 10 coefficients for each coefficient array: a[0] thru a[9] and b[0] thru b[9].
(3) You will need to use circular buffer for this design just like it is done in Example 5.4.
(4) Please comment all your software development.
(170 points)

- **IIR filter diagram**

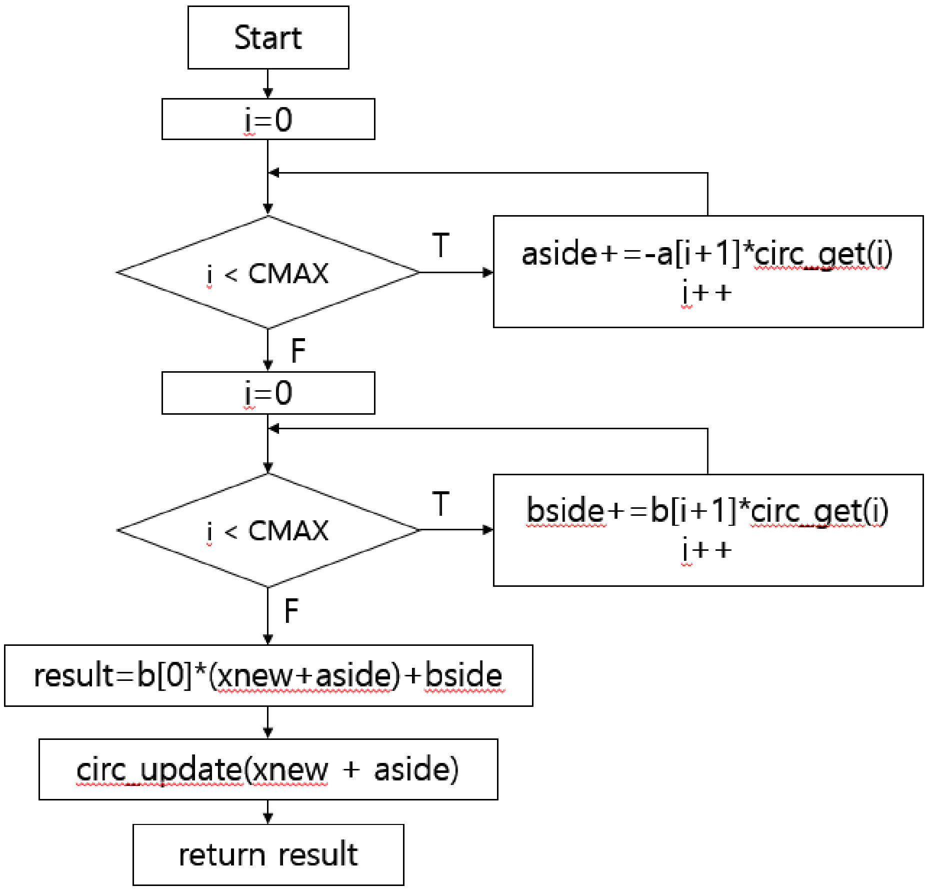

Start
i=0
i < CMAX
T
aside+=-a[i+1]*circ_get(i)
i++
F
i=0
i < CMAX
T
bside+=b[i+1]*circ_get(i)
i++
F
result=b[0]*(xnew+aside)+bside
circ_update(xnew + aside)
return result

Diagram for IIR2 filter

Byeongchan Gwak, Student ID : 501026

- Explanation for all of my software
  - Main() : Reads input sample values from INPUT array and calls iir2() & amplifier() functions.
  - iir() : This is a IIR filter funtion. It differs from FIR because it is using more coefficient. It mutiplies coefficient a and b with a sample value each. And then it adds up both values and return it for result.
  - amplifier(): This is a literally amplifier. It amplifies the value according to the input value.
  - circ_update() : Inserting a new value and then push off the head position by one.
  - circ_get() : Access circular array to get a sample value descending order because a most recent sample is at the top the circular array.
- Below is IIR implementation

```
#define INPUT_LEN   50 /* input sample length */
#define CMAX        10 /* filter order */

int INPUT[INPUT_LEN]   /* input sample */
int a[CMAX];     /* coefficient */
int b[CMAX];     /* coefficient */
int x[CMAX];     /* circular buffer for sample */
int y[CMAX];     /* output for after fir */
int z[CMAX];     /* output for after amplifier */
int pos;         /* position of current sample */

/* main function for iir2 & amplifier */
void main() {
    circ_init();
    int n;
    /* call iir2() & amplifier() for each input sample value */
    for(n=0; i<INPUT_LEN; n++) {
        y[n] = iir2(INPUT[n]);
        z[n] = amplifier(y[n]);
    }
}

/* initialize buffer */
void circ_init() {
    int i;
    for(i=0;i<CMAX;i++) {
        x[i] = 0;
        y[i] = 0;
        z[i] = 0;
    }
    pos = CMAX-1;
}
```

Byeongchan Gwak, Student ID : 501026

```c
/* given a new sample value, update the buffer and compute the IIR2 output */
int iir2(int xnew) {
    int i, aside, bside, result;

    circ_update(xnew);
    /* Calculate aside values */
    for(i=0, aside=0; i<CMAX; i++) { aside += -a[i+1] * circ_get(i); }
    /* Calculate bside values */
    for(i=0, bside=0; i<CMAX; i++) { bside += b[i+1] * circ_get(i); }
    result = b[0] * (xnew + aside) + bside;

    return result;
}

/* amplify the result of IIR filter */
int amplifier(int iir) {
    int result = iir * 150;
    return result;
}

/* add a new value and push off the oldest one */
void circ_update(int xnew) {
    /* the position moves from 0 to CMAX-1 */
    pos = ((pos == CMAX-1) ? 0 : (pos+1));
    x[pos] = xnew;
}

/* get the ith value from the buffer */
int circ_get(int i) {
    int ii = (pos-i) % CMAX;
    /* when result of modulo goes down minus
       add CMAX to make circular buffer */
    if(ii < 0) ii += CMAX;
    return x[ii];
}
```