Byeongchan Gwak, Student ID : 501026

Washington University in St.Louis

McKelvey School of Engineering

Fall Semester 2021

CSE467M: Embedded Computing Systems

Homework #2

Chapter 3 problems:

1) **Q3-3 - assembly code for ARM (30 points)**

```
        ADR r4, ds1
        LDR r3, [r4]        ; load ds1
        CMP r3, #0          ; compare to zero
        BNE TEST            ; not equal and then branch
TEST:
```

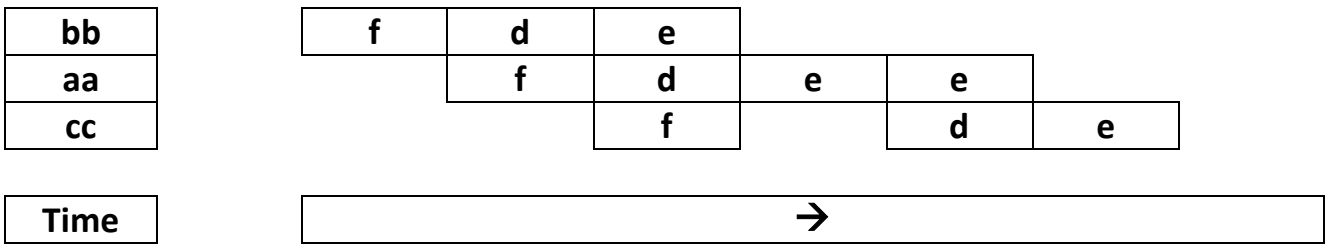
2) **Q3-4 - assembly code for ARM (30 points)**

```
        ADR r4, ds1
TEST:   LDR r3, [r4]        ; load ds1
        AND r3, r3, #1      ; bit operation with ds1
        CMP r3, #0          ; compare to zero
        BE TEST             ; if r3 is zero and then test again!
        ADR r4, dd1
        LDR r3, [r4]        ; load dd1
```

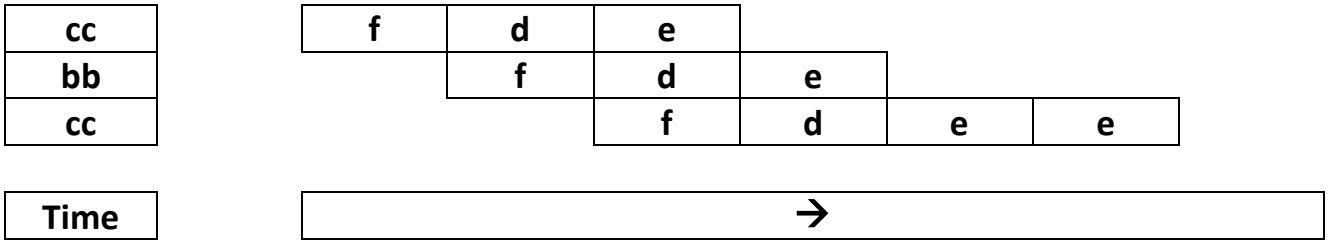
3) **Q3-23 (10 points)**
   a. Divide by zero
   b. illegal memory access
   c. undefined instructions
   d. resets
4) **Q3-24 (10 points)**
   a. A trap is like software interrupt and generates an exception condition. The most common use of the trap is to enter supervisor mode.
5) **Q3-26 (30 points)**
   a. Compulsory miss: It occurred when the first-time location is used. For example, starting a program for the first time.
   b. Capacity miss: It occurred by a too-large working set. For example, inside looping operations in a program.
   c. Conflict miss: It occurred when two locations map to the same location in the cache. For example, a program refers to a same cache memory block more than 2 times in a system using direct-mapped cache.
6) **Q3-27 (10 points)**
   a. 3ns * 0.96 + 70ns * 0.04 = 5.68ns
7) **Q3-28 (10 points)**
   a. Hit ratio : A
   b. A * 5ns + (1-A)*80ns = 6.5ns
   c. 75a = 73.5
   d. A = 0.98
   e. 98% cache hit rate should be achieved
8) **Q3-36 (30 points)**

- **F stands for fetch, d decode, e, execute**

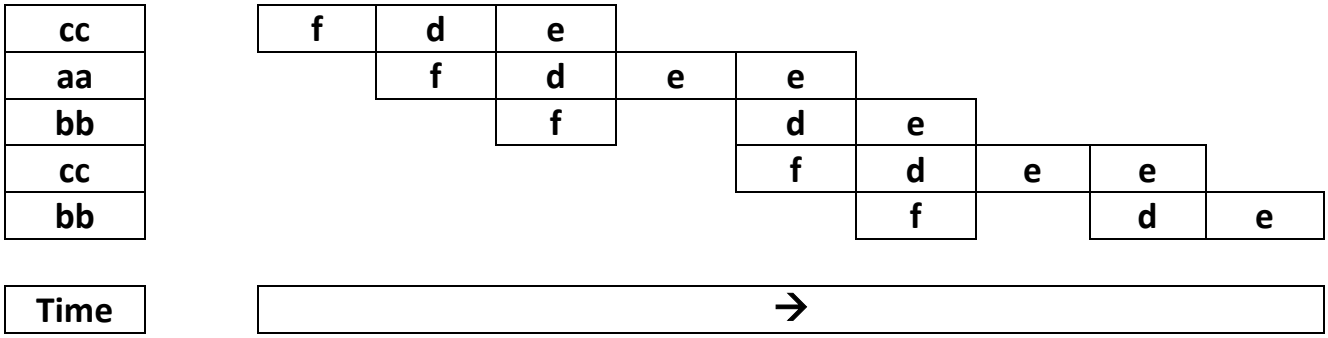
a. bb, aa, cc;



b. cc, bb, cc;



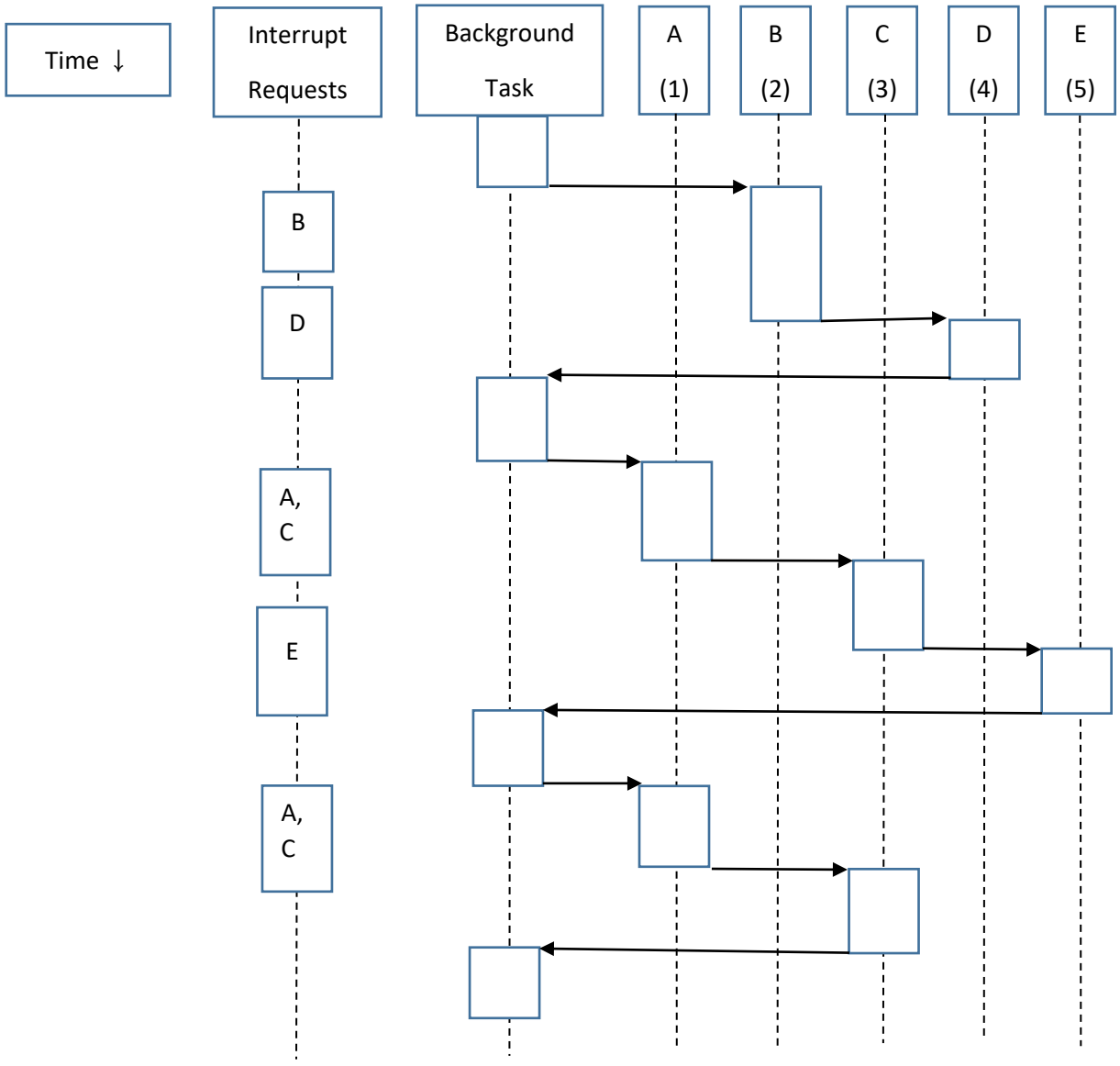
c. cc, aa, bb, cc, bb;



9) **Assume that we have embedded system with five I/O devices and the priority access levels for these 5 devices are A – priority 1 (highest), B – priority 2, C – priority 3, D – priority 4, E – priority 5 (lowest). Draw the UML sequence diagram just like one in Example 3.7 (textbook page 113) where the interrupt requests are happening as shown below:**

Background task is being executed for a while -> B interrupt occurs -> D interrupt occurs before B I/O device service is finished -> D I/O device service is finished and there is no interrupts for a while -> A,C interrupts happen at the same time -> E interrupt occurs before C I/O device service is finished -> E I/O device is finished and there is no interrupts for a while -> A,C interrupts happen at the same time -> After A and C I/O devices services are finished there is no interrupts for a while.

(60 points)



10) Memory contents using a 3-bit memory address scheme is shown below:

Address	Data
000	0100
001	0001
010	1111
011	0000
100	1000
101	1001
110	1100
111	0101

Apply the following pattern of addresses 000, 011, 100, 111, 110, 001, 101, 010 and show step-by-step the state of the cache after each memory access for:

(a) Direct-mapped cache with 4 blocks
(b) Two-way (banks) with 4 blocks each set associative cache with least-recently replacement (LRU) policy
(c) Two-way (banks) with 2 blocks each set associative with least-recently replacement (LRU) policy.

(150 points)

Byeongchan Gwak, Student ID : 501026

(a) Direct-mapped cache with 4 blocks

1.After 000 access:

Block	Tag	Data
00	0	0100
01	-	-
10	-	-
11	-	-

2.After 011 access:

Block	Tag	Data
00	0	0100
01	-	-
10	-	-
11	0	0000

3.After 100 access:

Block	Tag	Data
00	1	1000
01	-	-
10	-	-
11	0	0000

4.After 111 access:

Block	Tag	Data
00	1	1000
01	-	-
10	-	-
11	1	0101

5.After 110 access:

Block	Tag	Data
00	1	1000
01	-	-
10	1	1100
11	1	0101

6.After 001 access:

Block	Tag	Data
00	1	1000
01	0	0001
10	1	1100
11	1	0101

7.After 101 access:

Block	Tag	Data
00	1	1000
01	1	1001
10	1	1100
11	1	0101

8.After 010 access:

Block	Tag	Data
00	1	1000
01	1	1001
10	0	1111
11	1	0101

Byeongchan Gwak, Student ID : 501026

(b) Two-way (banks) with 4 blocks each set associative cache with least-recently replacement (LRU) policy

1.After 000 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	0100		-
01	-			-
10	-			-
11	-			-

2.After 011 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	0100		-
01	-			-
10	-			-
11	0	0000		-

3.After 100 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	0100	1	1000
01	-			-
10	-			-
11	0	0000		-

4.After 111 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	0100	1	1000
01	-			-
10	-			-
11	0	0000	1	0101

5.After 110 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	0100	1	1000
01	-			-
10	-		1	1100
11	0	0000	1	0101

6.After 001 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	0100	1	1000
01	0	0001		-
10	-		1	1100
11	0	0000	1	0101

7.After 101 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	0100	1	1000
01	0	0001	1	1001
10	-		1	1100
11	0	0000	1	0101

8.After 010 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
00	0	0100	1	1000
01	0	0001	1	1001
10	0	1111	1	1100
11	0	0000	1	0101

Byeongchan Gwak, Student ID : 501026

(c) Two-way (banks) with 2 blocks each set associative with least-recently replacement (LRU) policy.

1.After 000 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	00	0100	-	-
1	-	-	-	-

2.After 011 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	00	0100	-	-
1	01	0000	-	-

3.After 100 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	00	0100	10	1000
1	01	0000	-	-

4.After 111 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	00	0100	10	1000
1	01	0000	11	0101

5.After 110 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	11	1100	10	1000
1	01	0000	11	0101

6.After 001 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	11	1100	10	1000
1	00	0001	11	0101

7.After 101 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	11	1100	10	1000
1	00	0001	10	1001

8.After 010 access:

Block	Bank 0 tag	Bank 0 data	Bank 1 tag	Bank 1 data
0	11	1100	01	1111
1	00	0001	10	1001

11) Implement the following C for statement using ARM instruction set.

for (i=0, f=0; i<5; i++)
f = f + (a[i] + b[i])*x[i] – y[i];

Please comment all instructions in your assembly code.

After you implement assembly code perform execution time analysis and define a formula for the total execution time of the loop in cycles as it was done in Example 3.10 in the textbook on pages 134 thru 136.

(100 points)

```
.INIT:
        mov   r1, #0          ; set r1 for f to 0
        mov   r2, #0          ; set r2 for i to 0
        mov   r3, #0          ; separate index of arrays
        mov   r4, #5          ; set r4 for constant number 5
        adr   r5, a           ; load r5 with address of base of a array
        adr   r6, b           ; load r6 with address of base of b array
        adr   r7, x           ; load r7 with address of base of x array
        adr   r8, y           ; load r8 with address of base of y array

.TEST:
        cmp   r2, r4          ; compare i with number 5
        bge   BODY            ; if i >= 5, exit loop

.BODY:
        ldr   r9, [r5, r3]    ; get value of a[i]
        ldr   r10, [r6, r3]   ; get value of b[i]
        add   r0, r9,r10      ; a[i] + b[i]
        ldr   r9, [r7, r3]    ; get value of x[i]
        mul   r0, r0,r9       ; (a[i] + b[i]) * x[i]
        ldr   r9, [r8, r3]    ; get value of y[i]
        sub   r0, r0, r9      ; (a[i] + b[i]) * x[i] - y[i]
```

```
        add    r1, r1, r0          ; f = f + (a[i] + b[i]) * x[i] - y[i]

.UPDATE :
        add    r3, r3, #4          ; add one word offset to array index
        add    r2, r2, #1          ; i = i + 1
        b      .TEST
.LOOPEND:
```

Block	# of instructions	# of cycles
INIT	**8**	**8**
TEST	**2**	**2 best, 4 worst case**
BODY	**8**	**8**
UPDATE	**3**	**3**

Tloop = Tinit + N * (Tbody + Tupdate + Ttest_best) + Ttest_worst
= 8 + 5 * (8 + 3 + 2) + 4
= 8 + 5 * 13 + 4
= 8 + 65 + 4
= 77

12) A CMOS CPU is using 2.5 V voltage supply rail and clocking frequency of 100 MHz.

(a) If everything else stays the same and CPU voltage is dropped to 1.8V what % in power savings we can expect?
(b) If everything else stays the same and CPU clock frequency is dropped to 80MHz what % in power savings we can expect?
(c) If estimated dynamic power consumption is constant at 150mW and estimated static power consumption is constant at 20mW calculate total energy consumption for 1 hour time interval.

(30 points)

(a) P1 / P2 = $V_1^2Cf/V_2^2Cf = 2.5^2/1.8^2 \approx 1.92$**,**
A. 1/1.92 = 0.52, Therefore, we can expect 48% power savings
(b) F1 / F2 = 100 / 80 = 1.25, 25% power savings
A. 1/1.25 = 0.8, Therefore, we can expect 20% power savings
(c) Total energy consumption = $\int_0^t (Pdaynamic + Pstatic)dt$**=150 + 20 = 170mW**