Master - July 08

Solutions for E1.9/E2.19 (A)

A=analysis, D=design, C=calculated solution using taught theory, B=bookwork
NB - marking will be 1 mark for every 2% indicated on question paper (max 50 marks) so marks here are half of marks on question paper.

Solution to Question 1

36 minutes for the question => 9 minutes each part

a)
i) 1074.75
ii) 270F
iii) 112
iv) -1
v) 1023
[1 mark each, except iv & v 1 mark if both right]

[4C]

b)
seee eeee emmm mmmm mmmm mmmm mmmm mmmm

(i) s=0, e=158, m=0 => exp=31, mant=1 => 2^{31}=2147483648 **[2 marks]**
(ii) s=1,e=125,m= 10001....=>exp=2^{-2}, mant=1.001 =>-1.125/4=0.2812 **[2 marks]**

Unsigned exponent => 31 bit unsigned comparison can be used onfloating point, and combine with sign bit **[2 marks]**

[6C/A]

c)

R0=2
R1=4
R2=2-32=-30
R3=6
R4=0

Timing:
0: 0-3
4: 1-4
8: 2-6
C: 4-7

[4A]

d)
(i) -28*x
(ii)
RSB R1, R0, R0 lsl 3; R1 := R0*7
RSB R1, R1, R1 lsl 4; R1 := R1*15
(iii)
floor((2^{31}-1)/105) = 20452225 **(allow approximations)**

[6A/D]

Solution to Question 2

27 minutes for the question

This tests ability to understand low-level operation of ARM assembler instructions

For each part, deduct 1 mark for each column wrong down to minimum of 0 marks, except in allow consequent errors in memory write data. Assume mem[] = mem_{32}[]. Ignore entries in n/a columns

	r0	r1	r2	r3	r4	NZCV	Memory
a)	-2	3	&100 256	1	-3 &FFFFFFFD	1000	mem_{32}[&104] = 3 260
b)	&10B 267	1	&01020304 16909060	1	n/a	n/a	mem_8[&1] = 1
c)	&FC 252	$2^{31}+2^8$ &80000100 2147483904	2^9 &200 512	$2^{10}+1$ &401 1025	&01000000 2^{24} 16777216	0011	Mem_{32}[&100]=&401

[5A+5A+5A]

Solution to Question 3

27 minutes for the question

This questions tests understanding of the operation of different types of direct-mapped caches.

a)

each line has two words

i	tag	index	word sel	type
1	0	0	0	M
2	0	0	1	H
3	0	1	0	M
4	0	1	1	H
5	1	0	0	M

14,18,14,10, c, 8, 4, 0, 4, 8, c
H, M,H, H, M,H, M,H, H,H,H,

[5A]

b)

i	tag	index	word sel	type
1	0	0	0	M
2	0	1	0	M
3	0	2	0	M
4	0	3	0	M
5	1	0	0	M

M,M,H,H,H,M,M,M,H,H,H

[5A]

c)

1 R0,R4
2 none
3 R8, RC
4 none
5 W0,W4, R10,R14
6 none
7 none
8 none
9 W10,W14,R0,R4

[5A]

Solution to Question 4

This question tests whether the student understands the ARM conditional instructions and pipeline, and implications for code timing.

27 minutes for the question

```
TEST
        CMP R1, R0
        CMPEQ R3, R2
        BEQ T1
        RSB R5, R4, #0
        B T2
T1
        MOV R5, R4
T2
```

a)
if (R1=R0) and (R3=R2) then R5 := R4 else R5 := -R4

Either the BEQ branch is executed or the B branch is taken. The taken branch has execution time 4 cycles, all other instructions one cycle.

[4A]

b)

```
TEST
        CMP R1, R0
        CMPEQ R3, R2
        BEQ T1
        RSBNE R5, R4, #0
        MOVEQ R5, R4
```

[4D]

c) F=Fetch, D=Decode, E=execute

1F	1D	1E						
	2F	2D	2E					
		3F	3D	3E				
			4F (aborted)	S	S	4F	4D	4E

[4B]

d)
100MHz * 1/(1+BPL) = 100/(1+0.25*0.4*5)=66.6MHz. Must assume branch latency = pipeline length

[3C]

E1.9 – section B: Operating Systems
Sample model answers to exam questions 2008

Question 1

(a) [bookwork] The MQS scheduling algorithm divides ready processes into separate queues (for example, separate queues for foreground (interactive) and background (batch) processes); each queue has a priority associated with it, and queues with higher priority are given more time-slices for each of their processes. Each queue can have a separate algorithm for scheduling within the queue. Advantages: Flexible – allows fine control of scheduling. Disadvantages: Does not allow for the possibility of processes that change requirements throughout their execution (e.g. a process that started with a long CPU burst, but requires interaction later). The MFQS follows the same principles but allows for processes to move between queues, depending on their CPU utilisation, i.e. uses feedback to guide the redistribution to different queues. This allows for an additional layer of flexibility to MQS, and removes the disadvantage mentioned above for MQS. **[4]**

(b) [Bookwork]
Race conditions are situations in interprocess sychronisation where two or more processes are reading or writing some shared data, and the final result depends on who runs precisely when.
To ensure that the critical section of process A will always be executed before the critical section of process B, initialise a semaphore to 0, and require process B to wait on it until A signals that it is complete:

```
Init(Sem, 0)
Process A:                  Process B
....                        ....
Critical region             wait(Sem);
Signal(sem);                critical region;
End                         signal(Sem);
                            End
```
[4]

(c) [new computed example]

Optimal page replacement algorithm (6 page faults)

	5	**1**	**3**	**3**	**7**	**2**	**7**	**5**	**7**	**3**
Frame1	**5**	5	5		5	5				5
Frame2	**-**	1	1		7	7				7
Frame3	**-**	-	3		3	2				3

(the last replacement is not important; any will do) **[3]**

FIFO page replacement algorithm (7 page faults)

	5	**1**	**3**	**3**	**7**	**2**	**7**	**5**	**7**	**3**
Frame1	**5**	5	5		7	7		7		3
Frame2	**-**	1	1		1	2		2		2
Frame3	**-**	-	3		3	3		5		5

[3]

LRU (Least recently used) page replacement algorithm (7 page faults)

	5	**1**	**3**	**3**	**7**	**2**	**7**	**5**	**7**	**3**
Frame1	**5**	5	5		7	7		7		7
Frame2	**-**	1	1		1	2		2		3
Frame3	**-**	-	3		3	3		5		5

[3]

(d) [Bookwork]

The circular wait condition can be avoided by making sure that there can't be a circle in the resource allocation graph. In order to do so, you could impose as condition that only one resource can be allocated to a process at a time but this is not acceptable. A better method is to provide global numbering for all resources in the system – processes can request resources whenever they want but all requests must be made in numerical order. **[3]**

QUESTION 2:

(a) [bookwork]

First-fit: allocate first memory hole that is big enough. Advantages: fast allocation method. Disadvantages: can be very inefficient

Best-fit: allocate the smallest hole that is enough. Advantages: less inefficient in terms of space that first-fit. Disadvantages: tends to produce lots of remaining tiny fragments, and requires search through the entire list of memory holes

Worst fit: allocate the largest hole that is available. Advantages: after allocating the request, the remainder of that hole might still be usable. Disadvantages: requires search through the entire list of holes **[3]**

(b) [Bookwork]

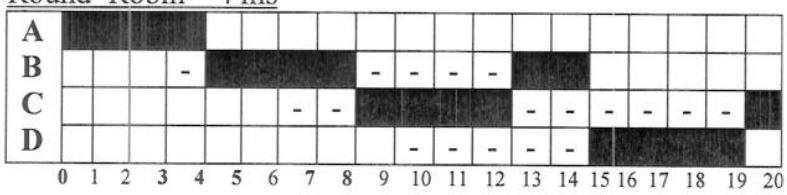
```
Procedure Init(var S: semaphore, Number: integer)
  s.count = Number;   s.queue = EmptyList;
end
```

```
Procedure wait(var s: semaphore)
s.count = s.count – 1;
if s.count < 0 then
        add process in s.queue
block process;
end;
```

```
Procedure signal(var s:semaphore)
s.count = s.count+1;
if s.count <= 0 then
  remove next process from s.queue
  place process in the ready queue
end;
```

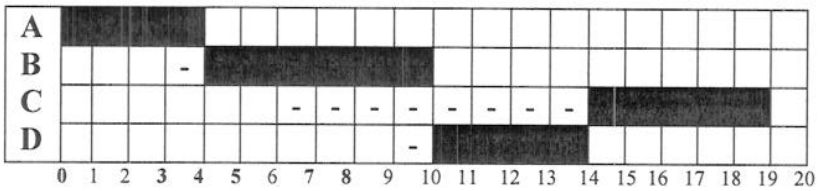
[3]

(c) Round Robin – 4 ms



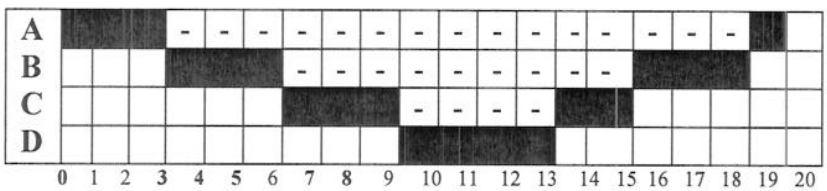
Avg waiting time: (0+5+8+5) / 4 = 4.5 ms, Avg turnaround time: (4+11+13+9)/4 = 37/4 = 9.25 ms **[3]**

Priority Scheduling (without preemption)



Avg waiting time: (0+1+8+1) / 4 = 2.5 ms, Avg. turnaround time: (4+7+13+5) / 4 = 29/4 = 7.25 ms **[3]**

Priority Scheduling (with preemption)



Average waiting time: (15+9+4+0) / 4 = 7 ms
Average turnaround time: (19+15+9+4) / 4 = 47 / 4 = 11.75 ms **[3]**

(d) [New computed example]

(i) The current state is safe since there exist safe sequences through which resources can be allocated without the possibility of a deadlock: e.g D->C->B->A. (more exist) **[3]**

(ii) (a) Request denied since, following the assignment of 3 resources to A, the new state will be unsafe. **[2]**