SRN | | | | | | | | | | | | |

PES University, Bangalore
(Established under Karnataka Act No. 16 of 2013)

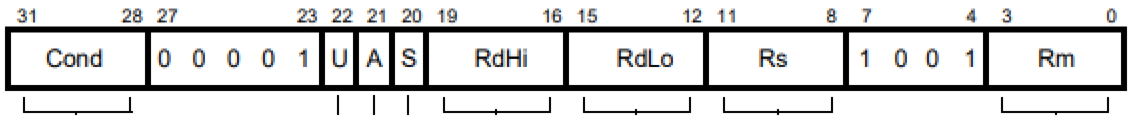
UE18CS243

END SEMESTER ASSESSMENT (ESA) B. TECH III SEMESTER- MAY 2020

UE18CS253 – Microprocessor and Computer Architecture

Time: 3 Hrs　　Answer All Questions　　Max Marks: 100

Note: All answers must be precise and to the point.

1.	a)	i. Explain why RSB instruction is required when SUB is available in ARM? i.e SUB R4, R1,R2 is same as RSB R4, R2,R1 Solution: If we want to perform R4=1-R2 SUB cannot be used as 1st operand cannot be immediate value, however RSB R4, R2,#1 will do. ii. Explain the similarities & differences between the ARM instructions CMP R0,R1 & CMN R0,R1 Both use the contents of R0 & R1 to set the condition codes. CMP compares 0 & R0–R1, CMN compares 0 & R0+R1	2+2
	b)	What is the output of the following program? Write an equivalent program using ADD instruction .TEXT mov r0,#3 RSB R0,R0,R0,LSL #2 SWI 0X011 R0=9, it is the multiplication by 3 add r1,r1,r1,lsl #1	4
	c)	 31 28 27 23 22 21 20 19 16 15 12 11 8 7 4 3 0 Cond 0 0 0 0 1 U A S RdHi RdLo Rs 1 0 0 1 Rm i. Explain the usage of SMLAL instruction. ii. Give Binary equivalent of for SMLAL R0,R1,R2,R3 (Cond = 1110 for always) Solution: The multiply forms SMLAL take two 32 bit numbers and multiply them to produce a 64 bit result of the form RdHi,RdLo := Rm * Rs. The lower 32 bits of the 64 bit result are written to RdLo, the upper 32 bits of the result are written to RdHi Binary equivalent 1110 00001 010 0000 0001 0010 1001 0011	4

SRN

	d)	Consider the program and the stack, if R13 points to the address 5400 before the execution of program, what is the status of the stack after execution of the following program. Give the updated value of R13.	4

```
.DATA
A:.word 10,20,30
.TEXT
LDR R0,=A
LDMIA R0!,{R1-R3}
STMFD R13!,{R1-R3}
SWI 0X011
```

53F0		
53F4		
53F8		
53FC		
5400		
5404		
5408		
540C		

STACK

R13 will have 53F4

53F0	
53F4	10
53F8	20
53FC	30
5400	
5404	
5408	
540C	

	e)	Describe the sequence of events which take place after an FIQ interrupt occurs. How does the interrupt handler return control to the program which was running when the interrupt occurred? What are the facilities available in FIQ mode which are not there in the ordinary interrupt mode?	4

When an FIQ occurs,

• ARM copies the current program status register (CPSR) to the saved status register (SPSR) of FIQ mode.
• It adjusts the instr set bit in the CPSR to change over to the ARM instruction set (by clearing the T bit).
• It sets the mode bits in CPSR to indicate that it is now in the FIQ mode.
• It sets the I bit in CPSR to disable IRQ because FIQ has higher priority.
• Stores the return address in LR. • stores the FIQ exception vector in PC.

To return control to the foreground processing,

• The handler copies the saved program status register to CPSR. This automatically restores the state of the instruction set, mode and the interrupt disable bits I and F to their original values.
• The handler then copies the link register back to PC to resume normal operation.

The FIQ mode has access to dedicated registers R8 through R12, its own stack pointer R13, link register R14 and saved program status register SPSR. It can thus use registers R8 through R12 without having to save them. Registers R0 through R7 of the parent mode are available to receive any global data that is relevant. However, any modification to R0 through R7 must be done only after saving a copy on the stack and then these registers must be restored from the stack on return. FIQ also has a high interrupt priority.

SRN

2 a) Explain how pipelining can improve the performance of a given instruction mix. 3

Pipelining provides “pseudo-parallelism” – instructions are still issued sequentially, but their overall execution (i.e. from fetch to write back) overlaps
- Performance is improved via higher throughput
o An instruction (ideally) finishes every short CC

b) Show how the instructions will flow through the 5 stage pipeline processor without forwarding and each stage require 1 clock cycle. 6

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L1: LDR R1,[R4]															
LDR R2, [R4],#-4															
ADD R3,R1,R2															
STR R3,[R4]															
SUB R4,R4,#4															
BNEZ R4 L1															

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L1: LDR R1,[R4]	F	D	EX	M	W										
LDR R2, [R4],#-4		F	D	EX	M	W									
ADD R3,R1,R2			F	D	X	X	EX	M	W						
STR R3,[R4]				F	X	X	D	EX	X	M	W				
SUB R4,R4,#4							F	D	X	EX	M	W			
BNEZ R4 L1								F	X	D	X	X	EX	M	W

c) Consider a simple in-order five-stage pipeline with a two-cycle branch misprediction penalty and a single-cycle load-use delay penalty. For a specific program, 30% of the instructions are loads, 20% are branches, the remaining 50% of instructions are simple single-cycle ALU operations. 5
Half of the load instructions are followed immediately by a dependent instruction, and 75% of branches are predicted correctly. What is the average CPI of this program on this processor?

The CPI would be 1 plus an extra cycle for loads followed by dependent instruction (15% of instructions), an extra two cycles for mis-predicted branches (25% of 20%).
Thus, the CPI is 1 + (1 *0.15) + (2 * 0.25 * 0.20) = 1.25.

d) Consider a program consisting of two conditional branches. The program will be executed many times. Below are the outcomes of each branch for one execution of the program core (T for taken and N for not taken). 6

Branch 1: T-N-T-N-T-N-T-N
Branch 2: T-T-T-N-T-T-T-N-T

SRN

Assume that the behavior of each branch remains the same for each program execution. For dynamic branch prediction schemes, assume that each branch has its own prediction buffer and each buffer is initialized to the same state before each execution.

1 Bit Predictor, initialized to predict taken			
	Correct Prediction	Wrong Prediction	Accuracy
Branch 1			
Branch 2			
2 bit Predictor, initialized to weakly predict taken			
	Correct Prediction	Wrong Prediction	Accuracy
Branch 1			
Branch 2			

1 Bit Predictor, initialized to predict taken			
	Correct Prediction	Wrong Prediction	Accuracy
Branch 1	1	7	6/17
Branch 2	5	4	
2 bit Predictor, initialized to weakly predict taken			
	Correct Prediction	Wrong Prediction	Accuracy
Branch 1	4	4	11/17
Branch 2	7	2	

3. A Pentium 4 microprocessor has two 8 Kbyte, Level 1 caches – one for data and one for instructions. However, a design team is considering another option – a single, 16 Kbyte cache that holds both instructions and data.
Additional specs for the 16 Kbyte cache include:
- Each block will hold 32 bytes of data
- The cache would be 2-way set associative
- Physical addresses are 32 bits.
- Data is addressed to the word and words are 32 bits

a) How many blocks would be in this cache? 3

The cache holds 2^{14} bytes of data
- Each block holds 2^5 bytes
- Thus, there are $2^{14} / 2^5 = 2^9 = 512$ blocks

b) How many bits of tag are stored with each block entry? 3

- Index:
o # of sets: $1024 / 2 = 256 = 2^8$
o Therefore 8 bits of index are needed
- Offset:

		o # of words per block = 32 / 4 = 8 o $2^3 = 8$ o Therefore 3 bits of offset - Tag o 32 – 3 – 8 = 21 bits of tag Therefore, 21 bits of tag need to be stored in each block.	
	c)	Each instruction fetch means a reference to the instruction cache and 35% of all instructions reference data memory. With the first implementation: - The average miss rate in the L1 instruction cache was 2% - The average miss rate in the L1 data cache was 10% - In both cases, the miss penalty is 9 CCs For the new design, the average miss rate is 3% for the cache as a whole, and the miss penalty is again 9 CCs. Which design is better and by how much? Answer Miss penaltyv1 = (1)(.02)(9) + (0.35)(.1)(9) = .18 + .063 = 0.495 Miss penaltyv2 = (.03)(9) = 0.270 V2 is the right design choice	10
	d)	Explain the advantages and disadvantages (in 4-5 sentences or a bulleted list) of using a direct mapped cache instead of an 8-way set associative cache. A direct mapped cache should have a faster hit time; there is only one block that data for a physical address can be mapped to - The above “pro” can also be a “con”; if there are successive reads to 2 separate addresses that map to the same cache block, then there may never be a cache hit. This will significantly degrade performance. - In contrast, with a set associative cache, a block can map to one of 8 blocks within a set. Thus,if the situation described above were to occur, both references would be hits and there would be no conflict misses. - However, a set associative cache will take a bit longer to search – could decrease clock rate.	4
4.	a)	The average memory access time (AMAT) can be modeled using the following formula: *AMAT = Hit time + Miss rate * Miss penalty* Name and explain (briefly) one technique for each of the three components of the formula in order to decrease the average memory access time **Reducing Miss penalty:** - Multilevel caches: 1st level small, but at the speed of the CPU, 2nd level larger but slower - Critical word first: don’t wait until the entire cache-block has been load, Focus − Giving priority to read misses over writes − Prefetching − Non Blocking Cache − Early Restart **Reducing Miss rate:** - Larger cache block size - Larger caches	6

SRN

- Higher associativity
- Way prediction and pseudo-associative caches
- Compiler optimization

Reducing Hit time:

- Small and simple caches
- Avoiding address translation
- Pipelined cache access
- Trace caches

b) Hari's program is designed to be 90% parallel. What speedup can Hari expect on 10 processors? What would be the maximum speedup on an infinite number of processors? 4

Sp = 1 / (f + (1 - f)/p) = 1 / (0.1 + (.9)/10) = 1 / .19 = 5.26

max Sp = 1 / (.1 + .9/∞) = 1 / .1 = 10

c) Priya executes her program and determines that in a parallel execution on 100 processors, 5% of the time is spent in the sequential part of the program. Priya's problem size can increase with an increasing number of processors. What is the scaled speedup of the program on 100 processors? 4

Sp = 100 + (1 - 100) * .05 = 100 - 4.95 = 95.05

d) What is the average memory access time for the following memory system? 6

- Level 1 cache: 91% hit rate, 1-cycle access time.
- Level 2 cache: 98% hit rate, 15-cycle access time.
- Memory: 140-cycle access time.

Cache hits and misses both require some number of cycles to access the cache (the access time). You may assume that all memory accesses are hits in main memory.

Solution

We incur the access time for a cache regardless of whether it's a hit or a miss. So if we have an access that is not in cache at all, we first spend 1 cycle accessing L1 to find out that it's a miss. Then we spend 15 more cycles accessing L2 to find out that we've missed again, and finally 140 cycles accessing memory.

We have 3 cases…

L1 hit (0.91 probability): 1 cycle

L1 miss, L2 hit (0.09 * 0.98 probability): 1 cycle + 15 cycles

L1/L2 miss, memory hit (0.09 * 0.02 probability): 1 cycle + 15 cycles + 140 cycles

Altogether:

AMAT = 0.91 * 1 + 0.09 * (0.98*(1+15) + 0.02*(1+15+140)) = 2.602 cycles

No individual memory access will take 2.602 cycles, but this should be the average over all memory accesses.

5. a) Modern multicores often have on-chip L2 caches as well as L1 caches. It is common to implement something called the "inclusion property", which means that if a certain cache block is evicted from the L2 cache (eg., due to a conflict) then it also must be evicted from the L1 cache. In one sentence, why would this 3

property be useful?

Solution:

Coherence mechanisms don't have to check both the L2 and the L1 for existence of a cache block, only the L2.

b) Consider the following sequence of instructions. The table on the right represents the latency incurred by each instruction.

I1: LDR R1, [R0]		
I2: MUL R2, R1, R3	Instruction	Latency
I3: ADD R4, R2, R3	LDR	3
I4: LDR R1, R5	MUL	6
I5: ADD R3, R1, R2	ADD	2
I6: LDR R1, [R5]		
I7: ADD R3, R1, R3		

i. List all RAW, WAW and WAR hazards found in the code.

ii. Rename enough registers in the code to eliminate as many as dependency.

iii. How many cycles would be needed to execute the example code if the pipeline stalled only on true dependences (RAW hazards) found in the code? Show the instruction schedule?

3
2
4

i. List all RAW hazards found in the code.

Solution

RAW: I1 _ I2 (for R1)
I2 _ I3 (for R2)
I2 _ I5 (for R2)
I4 _ I5 (for R1)
I5 _ I7 (for R3)
I6 _ I7 (for R1)

ii. List all WAW and WAR hazards in the code

WAR: I2 _ I4 (for R1)
I2 _ I5 (for R3)
I3 _ I5 (for R3)
I5 _ I6 (for R1)

WAW: I1 _ I4 (for R1)
I4 _ I6 (for R1)
I5 _ I7 (for R3)

iii. Rename enough registers in the code to eliminate as many as dependency.

I1: LDR R1, [R0]
I2: MUL R2, R1, R3
I3: ADD R4, R2, R3
I4: LDR **R6,** R5
I5: ADD R3, **R6**, R2
I6: LDR **R7**, [R5]
I7: ADD R3, **R7**, R3

How many cycles would be needed to execute the example code if the pipeline stalled only on true dependences (RAW hazards) found in the code? Show the instruction schedule?

Cycle	Instruction
1	I1: LDR R1, [R0]
2	stall
3	stall
4	stall
5	I2: MUL R2, R1, R3
6	stall
7	stall
8	stall
9	stall
9	stall
10	stall
11	I3: ADD R4, R2, R3
12	I4: LDR R6, [R5]
13	stall
14	stall
15	stall
16	I5: ADD R3, R6, R2
17	I6: LD R7, [R5]
18	stall
19	stall
20	stall
21	I7: ADD R3, R7, R3
22	stall
23	stall
24	Execution complete

c) Unroll the following loop twice and perform register renaming. 8

```
Loop: ldr r5, [r1]
      ldr r4, [r2]
      sub r5, r5, r4
      add r6, r6, r5
      add r2, r2, #4
      add r1, r1, #4
      bne r1, r3, Loop
Loop: ldr r5, [r1] #4
      ldr r4, [r2] #4
      sub r5, r5, r4
      add r6, r6, r5
      ldr r7, [r1]
      ldr r8, [r2]
      sub r7, r7, r8
      add r6, r6, r7
      add r2, r2, #8
      add r1, r1, #8
      bne r1, r3, Loop
```