1. (a) A research student wants to improve the performance of a processor by considering an enhancement E1 that applies to 20% of the original instructions, and speeds each of them up by a factor of 8. His supervisor has some concerns about E1 and suggests an alternative enhancement E2. Enhancement E2, if applied to 35% of the original instructions, would achieve the same overall speedup as obtained using enhancement E1. Determine the factor of enhancement using E2. Also comment about the concern of the research student's supervisor.

(7 marks)

Answer

E1 = 20% = 0.2

S1 = 8

$$\text{Speedup (E1, S1)} = \frac{1}{(1 - E1) + \frac{E1}{S1}} = \frac{1}{(1 - 0.2) + \frac{0.2}{8}} = \frac{40}{33}$$

E2 = 35% = 0.35

$$\text{Speedup (E2, S2)} = \frac{1}{(1 - E2) + \frac{E2}{S2}} = \frac{1}{(1 - 0.35) + \frac{0.35}{S2}} = \frac{1}{0.65 + \frac{0.35}{S2}}$$

E2 speedup equivalent to E1,

$$\frac{1}{0.65 + \frac{0.35}{S2}} = \frac{40}{33}$$

$$\frac{33}{40} = 0.65 + \frac{0.35}{S2}$$

$$\frac{33}{40} = \frac{13}{20} + \frac{7}{20(S2)}$$

$$\frac{7}{40} = \frac{7}{20(S2)}$$

140(S2) = 280

S2 = 2

$$\text{For E1, maximum achievable speedup} = \frac{1}{1 - E1} = \frac{1}{1 - 0.2} = 1.25$$

$$\text{For E2, maximum achievable speedup} = \frac{1}{1 - E2} = \frac{1}{1 - 0.35} = 1.54$$

E2 speedup is better than E1.

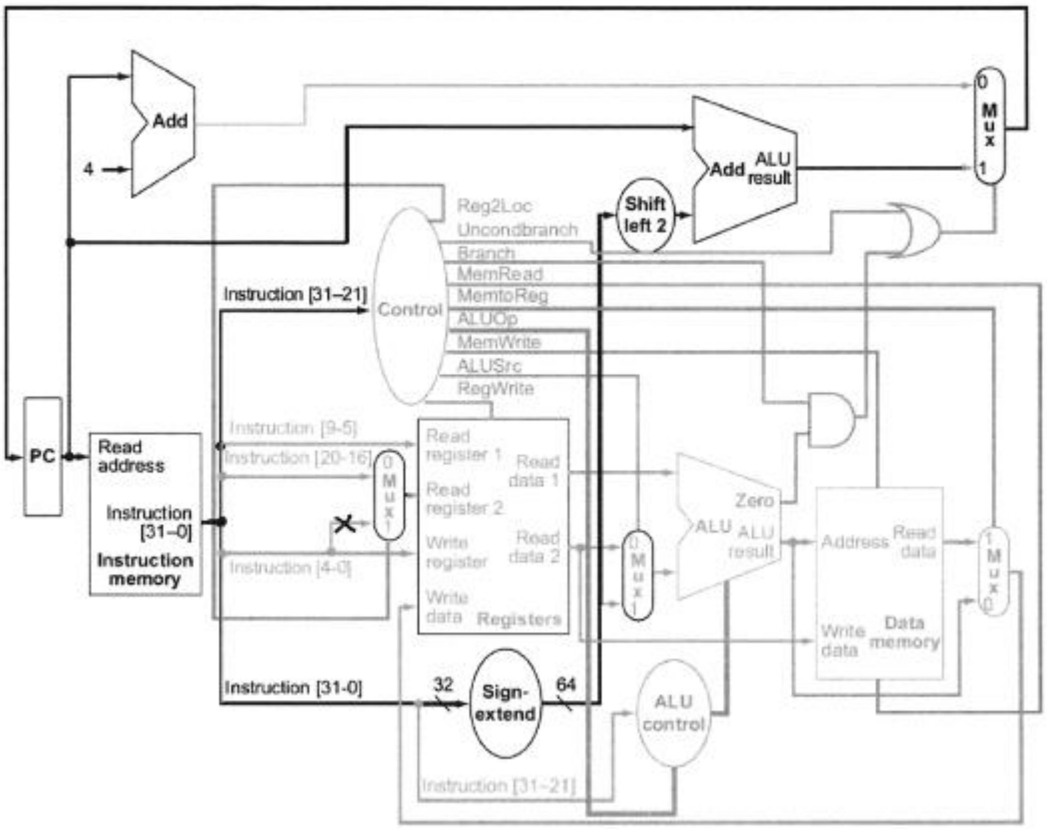
Hence, supervisor concern is valid, E2 enhancement should be used.

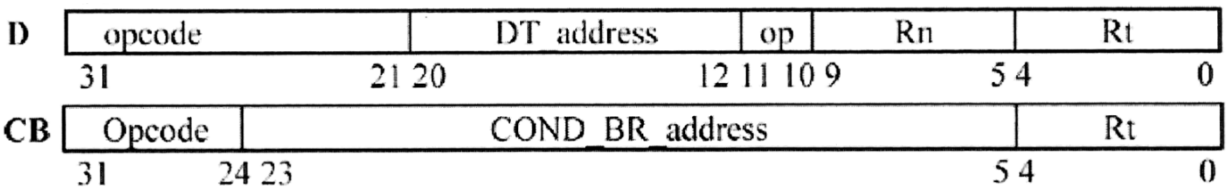
1. (b) Briefly explain the working of the instructions "LDUR X0, [X1, #8]" and "CBZ X0, #8". Figure Q1 shows the LEGv8 architecture, where one line is marked with "X". "X" indicates there is a hardware failure due to which that bus is broken. Indicate which of the instructions above will have issue in executing. Explain in detail the reason for the same.

(9 marks)

Answer

Figure Q1





D	opcode	DT_address	op	Rn	Rt
	31 21	20 12	11 10	9 5	4 0

CB	Opcode	COND_BR_address	Rt
	31 24	23 5	4 0

Working of the Instructions

"LDUR X0, [X1, #8]"

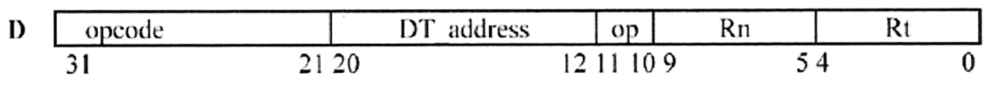
1. Read the data from register X1.
2. Compute the memory address by adding the values of register X1 to the immediate value “8” using the ALU.
3. Write the values from the computed memory address to the register X0.
4. [X0] ← mem[[X1] + [8]]

"CBZ X0, #8"

1. Read the data from register X0.
2. If values from X0 == 0, the PC is updated to the branch target address (current PC + #8<<2).
3. Otherwise, the next instruction is executed.
4. [new PC] ← PC + 8*4; if X0 = 0

Bus is broken

"STUR X0, [X1, #8]"



Instructions fetched:

Instructions [31-21] - opcode

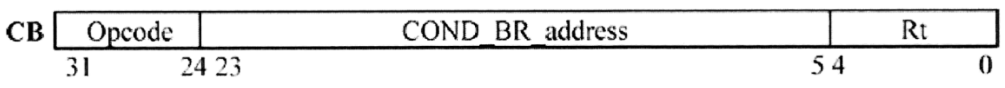
Instructions [20-12] - immediate value

Instructions [9-5] - read register 1 (X1)

Instructions [4-0] - write register (X0)

Since LDUR does not require to read register 2, hence LDUR has no issue.

"CBZ X0, #8"



Instructions fetched:

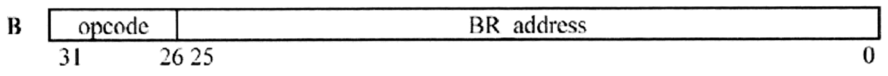
Instructions [31-24] - opcode

Instructions [23-5] - immediate value

Instructions [4-0] - read register 2 (X0)

As CBZ requires to read the data from register X0, it is unable to read the register when the bus is broken. Hence, CBZ will have issue in executing the instruction.

1. (c) Determine the range, minimum and maximum possible values to which an instruction can unconditionally branch, given that the PC value of the branch instruction is 0x10FC. (9 marks)



Answer

No. of address bits for unconditional branch = 25 – 0 + 1 = 26 bits

The maximum positive number for 26 bits

= 01 1111 1111 1111 1111 1111 1111 (0x1FFFFF)

Maximum PC = PC + Sign extended (offset) << 2

= 0x10FC + (0x1FFFFFF)*4

= 0x080010F8

The maximum negative number for 26 bits

= 10 0000 0000 0000 0000 0000 0000 (0x2000000)

Minimum PC = PC + Sign extended (offset) << 2

= 0x10FC + (-0x2000000)*4

= 0xF80010FC

Range PC = 0xF80010FC to 0x080010F8

2. (a) Name the three different kinds of hazards that introduce penalty stall cycles in a pipelined architecture. Give one example for each kind of hazards.

(4 marks)

Answer

1. Structural hazard
   Resource conflict
   Example: Memory access by IF and MEM stage in the same cycle.

2. Data hazard
   Data dependency between instructions
   Example: “ADD X3, X1, X2” followed by “SUB X4, X3, X2” (RAW hazard).

3. Control Hazard
   Uncertainty in branch target address
   Example: Conditional branch “CBZ X0, #8” causing pipeline stalls until the branch decision is resolved.

2. (b) Listing Q2 shows a code segment that is intended to be executed in a 5-stage pipelined LEGv8 processor. The program counter is updated with the branch target address at the Execute stage. Let the initial values be `X7=0x0000001000000000` and `X8=0x0000000000001100` (CBZ: *branch on equal to zero*).

<u>Listing Q2</u>

```
I1   loop: LDUR  X0,  [X7,  #0]
I2         LDUR  X1,  [X8,  #0]
I3         XOR   X2,  X1,   X0
I4         STUR  X2,  [X7,  #0]
I5         SUBI  X7,  X7,   #8
I6         SUBI  X8,  X8,   #16
I7         CBZ   X8,  Finish
I8         B     loop


     Finish
```

(i) Calculate the steady-state CPI of the code segment in Listing Q2 with the help of a reservation table for the execution of the code if full data forwarding is allowed. Show the forwarded paths and the dependencies. Find the total number of loop iterations.

(8 marks)

2 (i) Answer

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
loop: LDUR X0, [X7, #0]	I1.	F	D	E	M	W											
LDUR X1, [X8, #0]	I2.		F	D	E	M	W										
XOR X2, X1, X0	I3.			F	D	S	E	M	W								
STUR X2, [X7, #0]	I4.				F		D	E	M	W							
SUBI X7, X7, #8	I5.						F	D	E	M	W						
SUBI X8, X8, #16	I6.							F	D	E	M	W					
CBZ X8, Finish	I7.								F	D	E	M	W				
B loop	I8.									S	S	F	D	E	M	W	
Finish																	

Data dependency:

RAW: I1, I3 at X0
I2, I3 at X1
I3, I4 at X2
I6, I7 at X8

WAR: I1, I5 at X7
I4, I5 at X7
I2, I6 at X8

Assuming the single instruction negligible

$$\text{Steady state CPI} = \frac{\text{No. of instructions + No. of stalls + No. of control}}{\text{No. of instructions}}$$

$$= \frac{8+1+2}{8} = \frac{11}{8} = 1.375$$

X8 = 0x0000000000001100 (hex)
= 4352 (ten)

$$\text{Total number of loop iterations} = \frac{4352}{16} = 272$$

2. (ii) The code segment shown in Listing Q2 is now intended to be executed in a two-way superscalar processor. In the superscalar processor, one way is exclusively for load and store instructions whereas the other way can execute all instructions except load and store. Find the CPI achieved by the superscalar architecture. Note that full data forwarding is allowed.

(9 marks)

Answer

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
loop: LDUR X0, [X7, #0]	I1.	F	D	E	M	W											
LDUR X1, [X8, #0]	I2.		F	D	E	M	W										
XOR X2, X1, X0	I3.			F	D	S	E	M	W								
STUR X2, [X7, #0]	I4.				F		D	E	M	W							
SUBI X7, X7, #8	I5.						F	D	E	M	W						
SUBI X8, X8, #16	I6.							F	D	E	M	W					
CBZ X8, Finish	I7.								F	D	E	M	W				
B loop	I8.									S	S	F	D	E	M	W	
Finish																	

	Way-1 (rest of instructions)	Way -2 (LDUR and STUR only)	Cycle
loop	nop	LDUR X0, [X7, #0]	1
	nop	LDUR X1, [X8, #0]	2
	SUBI X8, X8, #16	nop	3
	XOR X2, X1, X0	nop	4
	CBZ X8, Finish	STUR X2, [X7, #0]	5
	SUBI X7, X7, #8	nop	6
	B loop	nop	8

CPI after 2-way superscalar architecture

$$= \frac{7}{8}$$

$$= 0.875$$

(iii) Briefly comment on the methods that can reduce the CPI of the superscalar architecture depicted in Q2(b)(ii).

(3 marks)

Answer

Combined effect of superscalar architecture and instruction reordering helped us improved the performance of the system.

3. (a) Name three cache organization schemes and briefly describe their key difference in design.

(5 marks)

Answer

- Direct-mapped cache
  - a block can only be in one place
- Fully associative cache
  - blocks can go anywhere
- Set associative cache
  - a block can be in any entry of a set

(b) Figure Q3 depicts the memory access workflow of a byte-addressable machine.

(i) Calculate the missing value of **Y** for the L1 cache in Figure Q3, assuming that it is an eight-way set associative cache which contains 1024 cache blocks.

(3 marks)

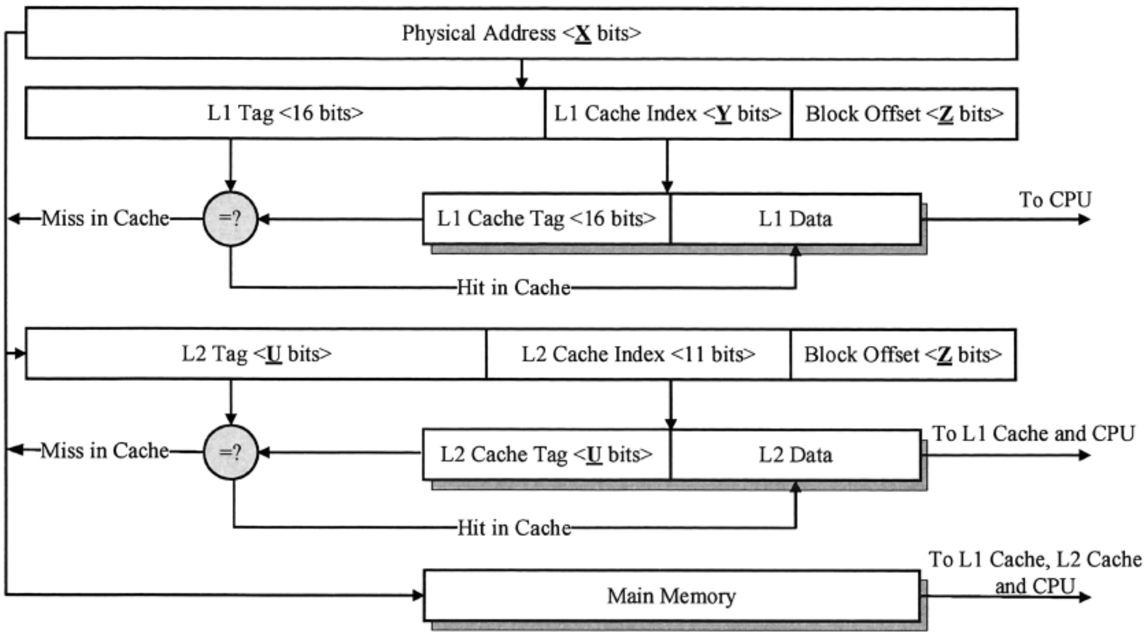


Figure Q3

Answer

No. of block = 1024

No. of way = 8

No. of set $= \dfrac{\text{No. of block}}{\text{No. of way}} = \dfrac{1024}{8} = 128$

Index Y = $\log_2$(No. of set) = $\log_2(128)$ = 7 bits

3. (b) (ii) Following Q3(b)(i), given that the size of the main memory is 2GB, is it possible to know the block size of the L1 cache in Figure Q3? Justify your answer.

(6 marks)

Answer

L1 Memory size	= 2 GB
L1 Address size	= $\log_2$(Memory size) = $\log_2$(2 * 1024 * 1024 * 1024) = 31 bits
L1 Tag	= 16 bits
L1 Index	= 7 bits
L1 Address size	= Tag + Index + Offset
L1 Offset	= Address Size – Tag – Index = 31 – 16 – 7 = 8 bits
L1 Block size	= 2^8 = 256 Bytes

(iii) Following Q3(b)(i) and Q3(b)(ii), what is the minimum size of the L2 cache in the number of bits, given that it is a two-way set associative cache in which each entry has enough storage for the valid, dirty and LRU (least recently used) status?

(6 marks)

Answer

L2 Address size	= 31 bits
L2 Offset	= 8 bits
L2 No. of way	= 2
L2 Valid	= 1 bit
L2 Dirty	= 1 bit
L2 Address size	= Tag + Index + Offset
L2 Tag	= Address Size – Index – Offset = 31 – 11 – 8 = 12 bits
L2 Index	= $\log_2$(No. of set) = 11 bits
L2 No. of set	= 2^{11} = 2048
L2 Data size	= Block size = 256 * 8 = 2048 bits

L2 Cache size = (Valid + Dirty + LRU + Tag + Data) * Way * Set
= (1 + 1 + 1 + 12 + 2048) * 2 * 2048
= 8450048 bits
= 1056256 Bytes
= 1031.5 KB

3. (b) (iv) The miss rates of the L1 and L2 caches are 2% and 1%, respectively. The times for the L1 cache, the L2 cache and the main memory accesses are 2, 25 and 200 cycles, respectively. What is the speedup in average memory access time (AMAT) by adding the L2 cache to the machine?

(5 marks)

Answer

L1 Miss rate	= 2% = 0.02
L2 Miss rate	= 1% = 0.01

L1 Cache	= L1 Hit time	= 2 cycles
L2 Cache	= L2 Hit time	= 25 cycles
Main memory	= Miss penalty	= 200 cycles

AMAT	= Hit time + Miss rate * Miss penalty	
L1 AMAT	= 2 + 0.02 * 200	= 6 cycles
L2 AMAT	= 25 + 0.01 * 200	= 27 cycles

L1 + L2 AMAT = L1 Hit time + L1 Miss rate * L2 AMAT
= 2 + 0.02 * 27
= 2.54 cycles

$$\text{Speedup} = \frac{\text{L1 AMAT}}{\text{L1+L2 AMAT}} = \frac{6}{2.54} = \frac{300}{127} = 2.36$$

4 (a) List the four types of computing models in Flynn's classification system. Which type of computing model does a typical GPU (Graphics Processing Unit) belong to? Briefly explain the reason.

(5 marks)

Answer

- Single instruction, single data stream – SISD
- Single instruction, multiple data stream – SIMD
- Multiple instruction, single data stream – MISD
- Multiple instruction, multiple data stream – MIMD

GPU belongs to SIMD. GPU consists of many SIMD processing cores that execute the same instruction but with different data.

(b) Briefly explain how the specifiers of `__global__` and `__shared__` are typically used in CUDA C programming, respectively.

(6 marks)

Answer

- `__global__` is used to declare a kernel function as device code
  - Called from the host

- `__shared__` is used to declare a variable/array in shared memory
  - Data is shared between threads in a block

(c) For the CUDA kernel configuration `kernel_A<<<4,128>>>(...)`, describe how the threads are to be launched and executed on a GPU, how the GPU resources are to be used, and how we can identify the individual threads during execution. (8 marks)

Answer

- Launch
  - The kernel will launch 4 thread blocks with 128 threads in each block, with a total of 4 * 128 = 512 threads.

- Execution
  - These threads are executed in parallel, and the execution is divided into thread blocks that are scheduled on GPU Streaming Multiprocessors (SMs).

- Warp
  - When a SM is given one or more thread blocks to execute, it first partitions the threads into group of 32 known as a Warp.

- Hardware Resource Allocation
  - On-chip resources are used to store the execution context for each warp processed by a SM
    - E.g. program counters, registers, shared memory, etc.

- Threads identification
  - Each thread is given a unique thread ID within the block (`threadIdx`).
  - Each thread block is given a unique block ID (blockIdx).
  - Identify a thread by combining variables `threadIdx` and `blockIdx` together with `blockDim`.

4. (d) Figure Q4 shows the code snippet of a CUDA program that consists of a kernel `kernel_B()` launched by the host. Identify a problem in the kernel code and briefly discuss how the problem will affect the performance of the program execution.

(6 marks)

```
__global__
void kernel_B(int n, int x){
  int i;
  i = n * 200;
  if (i < 300)
     x = n;
  else
     x = n - 300;
  }


int main(void){
     :
  kernel_B<<<2, 64>>>(d_n, d_x);
     :
     _
  return 0;
  }
```

Figure Q4

Answer

The problem lies in how the variable x is used in kernel_B().

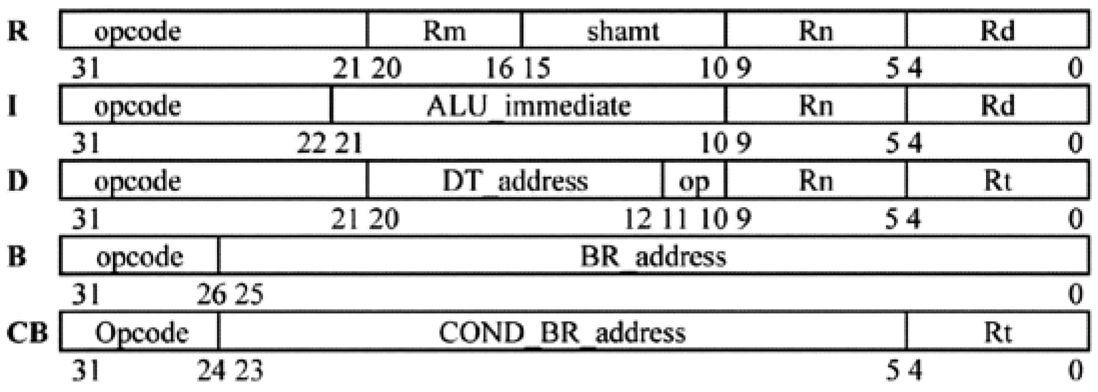
In the code, x is passed by value to the kernel and updated independently within each thread.

Since CUDA threads execute in parallel, changes to x do not affect the overall kernel behaviour because x is not a pointer to device memory.

Instead, x is treated as a local copy for each thread and is discarded after the kernel finishes execution.

To ensure x is updated correctly, it should be passed by reference or as a pointer.

Appendix – Instruction Formats


R
opcode
Rm
shamt
Rn
Rd
31 21 20 16 15 10 9 5 4 0
I
opcode
ALU_immediate
Rn
Rd
31 22 21 10 9 5 4 0
D
opcode
DT_address
op
Rn
Rt
31 21 20 12 11 10 9 5 4 0
B
opcode
BR_address
31 26 25 0
CB
Opcode
COND_BR_address
Rt
31 24 23 5 4 0