Indian Institute of Technology (IIT-Bombay)

AUTUMN Semester, 2026

COMPUTER SCIENCE AND ENGINEERING

CS230: Digital Logic Design + Computer Architecture

End Semester Examination

Full Marks: 55

Time allowed: 3.0 hours

INSTRUCTIONS: Do not make any extra assumptions other than those stated in the questions. The marks will only be given if a precise, sound, and clearly written explanation is provided that is understandable by the examiners. Any extra unnecessary writing, verbal arguments based on that, and unnecessary crib will result in a reduction of 5 marks.

Roll Number: _____

Name: _____

QN	1	2	3	4	5	6	7	8	9	10	Total
Marks											

1. Choose the correct option. A 1-bit branch predictor predicts the outcome observed during the previous execution of the same branch. The predictor is initially set to predict Not Taken (N). Suppose the actual outcomes of four consecutive executions of the branch are:

$$T,\ T,\ N,\ T$$

where T denotes Taken and N denotes Not Taken.

How many of the four branch predictions are correct? (1 mark)

(a) 0

(b) 1

(c) 2

(d) 3

Solution: (b) 1
No part marking

2. A processor has a direct-mapped data cache. A program repeatedly accesses two arrays, A and B, in an alternating fashion. Although both arrays are small enough to fit together in the cache, the program experiences a large number of cache misses due to conflicts between blocks of the two arrays mapping to the same cache lines.

   Which of the following modifications is most likely to improve cache performance without increasing the total cache capacity? Only one answer is correct. (2 marks)

   (a) Increase the cache line size while keeping the total cache capacity unchanged.

   (b) Make the cache associative(associativity=2) while keeping the total cache capacity unchanged.

   (c) Replace the write-back policy with a write-through policy.

   (d) Increase the processor clock frequency.

Solution: (b)
No part marking

3. Choose the correct statement(s). One or multiple statement(s) can be correct. **You will only be given marks if you choose all the correct and no incorrect options.** (2 marks)

   (a) Each row in DRAM has a row buffer.

   (b) Each bank in DRAM has a row buffer.

   (c) A DRAM row typically contains multiple cache lines.

   (d) None of the above is correct.

Solution: (b)(c)
No part marking

4. The octal representation of an integer is $(342)_8$. If the same bit pattern is interpreted as an 8-bit signed integer in two's-complement representation, what is its decimal value? (2 marks)

   (a) 226

   (b) -98

   (c) 76

   (d) -30

Solution: (d)
No part marking

5. A full adder has three single-bit inputs A, B, and C_{in}, and two single-bit outputs: sum S and carry-out C_{out}. Which of the following statements is always correct? Assume that + denotes ordinary integer addition, while $\oplus$ denotes XOR. Explain the correct option. (3 marks)
   (a) $S = C_{\text{out}}$
   (b) $S \oplus C_{\text{out}} = A \oplus B$
   (c) $S + 2C_{\text{out}} = A + B + C_{\text{in}}$
   (d) $C_{\text{out}} = AB$

Solution: (c)
A full adder adds three single-bit inputs A, B, and C_{in} and produces a two-bit result consisting of the carry-out C_{out} as the most significant bit and the sum S as the least significant bit.

Since the carry-out bit has a positional weight of 2, the arithmetic value of the output is

$$2C_{\text{out}} + S.$$

Therefore, the fundamental arithmetic relation for a full adder is

$$\boxed{S + 2C_{\text{out}} = A + B + C_{\text{in}}}$$

which holds for every possible input combination.
Give 2 if the relation $2C_{out} + S$ is identified. Give 3 only if everything is correct.

6. For the following questions, assume the following system configuration:
   - 28-bit virtual addresses.
   - 16 MiB of byte-addressable physical memory, with LRU page replacement.
   - 4 KiB page size.
   - Fully associative TLB with 16 entries and an LRU replacement policy.
   - **Note:** 1 MiB = 2^{20} bytes; 1 KiB = 2^{10} bytes.

   (a) What is the VPN:PO (Virtual Page Number : Page Offset) breakdown for virtual memory? (2 marks)
   (b) What is the PPN:PO (Physical Page Number : Page Offset) breakdown for physical memory? (3 marks)
   (c) Assume:

- Two non-overlapping, page-aligned character arrays pointed to by `src` and `dst`. That is, both arrays start at page boundaries and occupy distinct virtual pages. Their page ranges are disjoint.
- Assume `strlen(dst)` $\geq 20 \times 4096$.
- `strlen(dst)` $\leq$ `strlen(src)`.
- The entire code fits within a single page.
- The code page is always resident in physical memory.
- The TLB initially contains a translation for the code page.
- Ignore memory accesses associated with loop control, local variables, and the array `rand`.
- Assume that instruction fetches keep the code-page translation in the TLB throughout execution. The remaining entries may contain arbitrary valid translations or may be empty.
- This is the only process running on the system.

Consider the following code:

```
// Let src and dst be char*.
// Assume rand is an array of integers initialized
// with random integers in [0, strlen(dst) - 1],
// without repeats.
// Assume max_replacements <= strlen(dst)
// and is sufficiently large for the questions below.

for (i = 0; i < max_replacements; i++) {
    dst[rand[i]] = src[rand[i]];
}
```

Answer the following questions:

(i) How many page faults would occur after or an arbitrary group of 20 consecutive iterations of the loop in the **best case** and the **worst case**? Assume that the initial contents of physical memory can vary between the best and worst cases. Clearly describe the initial memory state and the sequence of memory accesses that produce each case. (5 marks)

(ii) How many complete iterations of the loop can execute before a TLB miss occurs in the **best case** and the **worst case**? You must assume the loop runs long enough to encounter a miss. Clearly write down what is the best case and what is the worst case. (5 marks)

Solution:

(a) **VPN:PO breakdown** [2 marks]

The virtual address is 28 bits wide, and the page size is 4 KiB = 2^{12} bytes.

Therefore, the page offset requires

$$PO = \log_2(2^{12}) = 12 \text{ bits.}$$

The virtual page number requires

$$VPN = 28 - 12 = 16 \text{ bits.}$$

Hence,

$$\boxed{\text{VPN:PO} = 16 : 12}$$

(b) **PPN:PO breakdown** [3 marks]

The physical memory size is

$$16 \text{ MiB} = 16 \times 2^{20} = 2^{24} \text{ bytes.}$$

Since the memory is byte-addressable, the physical address is 24 bits wide.

The page offset requires 12 bits.

Therefore,

$$PPN = 24 - 12 = 12 \text{ bits.}$$

Hence,

$$\boxed{\text{PPN:PO} = 12 : 12}$$

(c) **Memory access analysis**

Each iteration of the loop performs two data-memory accesses:

- One read from `src[rand[i]]`.
- One write to `dst[rand[i]]`.

Since both arrays are page-aligned, the same index refers to corresponding page numbers (relative to the beginning of each array).

Furthermore, the arrays occupy disjoint virtual pages, so the two accesses require separate page translations.

(i) **Page faults during 20 consecutive iterations** [5 marks]

Best case:

Assume that all pages referenced during the selected 20 iterations are already resident in physical memory.

In this case, none of the memory accesses causes a page fault.

Therefore,

$$\boxed{\text{Best case} = 0 \text{ page faults}}$$

Worst case:

Assume that:

- Each of the 20 iterations accesses a different page of `src` and a different page of `dst`.
- None of these 40 pages is initially resident in physical memory.

Since each iteration accesses two previously nonresident pages, it causes two page faults.

Thus,

$$\text{Page faults} = 20 \times 2 = 40.$$

Therefore,

$$\boxed{\text{Worst case} = 40 \text{ page faults}}$$

This is feasible because the physical memory contains

$$\frac{2^{24}}{2^{12}} = 4096$$

page frames, which is sufficient to hold the 40 data pages and the code page.

(ii) **Iterations before the first TLB miss** [5 marks]

The fully associative TLB contains 16 entries.

One entry is occupied by the code-page translation, which remains in the TLB throughout execution.

Therefore, the number of entries available for data-page translations is

$$16 - 1 = 15.$$

Best case:

Assume that the TLB initially contains the translations for the pages that will be accessed by the loop.

Since the arrays are page-aligned and occupy disjoint virtual pages, accessing a particular 4 KiB range of indices requires two translations: one for `src` and one for `dst`.

Therefore, the maximum number of complete source–destination page pairs that can be accommodated in the TLB is

$$\left\lfloor \frac{15}{2} \right\rfloor = 7.$$

Each page pair allows access to 4096 distinct character indices without a TLB miss.

Since the indices in `rand` do not repeat, the maximum number of iterations before the first TLB miss is

$$7 \times 4096 = 28672.$$

This maximum is achieved by accessing all 4096 distinct indices from each of the seven cached page pairs before accessing an index belonging to an uncached page pair.

Therefore,

$$\boxed{\text{Best case} = 28672 \text{ complete iterations}}$$

Worst case:

Assume that the TLB initially contains only the code-page translation, or that the remaining entries do not contain the translations needed by the first iteration.

The first iteration accesses `src[rand[0]]` and `dst[rand[0]]`.

Since the required translations are absent, a TLB miss occurs during the first iteration.

Hence, no complete iteration executes before the first TLB miss.

Therefore,

$$\boxed{\text{Worst case} = 0 \text{ complete iterations}}$$

No partial marks for (a),(b). For (c).i, (c)ii. Give 1 if best/worst case is correct but no correct explanation. Give 2.5 if best/worst case is correct with correct explanation.

7. Consider a system with a three-level cache hierarchy (L1, L2, and L3) and the following specifications:

   - For every 1000 CPU-to-memory references:
     - 40 references miss in L1 cache.
     - 20 references miss in L2 cache.
     - 10 references miss in L3 cache.
   - L1 cache hit time = 1 clock cycle.
   - L2 cache hit time = 10 clock cycles.
   - L3 cache hit time = 100 clock cycles.
   - Main memory access time = 400 clock cycles.
   - Average memory references per instruction = 1.25.
   - Ideal CPI (including L1 hit time) = 1.

   Assume that the caches are accessed sequentially (L1 $\rightarrow$ L2 $\rightarrow$ L3 $\rightarrow$ Main Memory), and the access times at successive levels are additive. Also assume memory accesses are blocking, memory stalls do not overlap, and there are no other pipeline stalls.

   Answer the following questions:

   (i) What is the local miss rate of the L2 cache? (1 mark)

(ii) What is the global miss rate of the L2 cache? (1 mark)

(iii) What is the local miss rate of the L3 cache? (1 mark)

(iv) What is the global miss rate of the L3 cache? (1 mark)

(v) What is the Average Memory Access Time (AMAT) considering all three levels of cache? (2 marks)

(vi) What is the effective CPI with memory stalls? (2 marks)

Solution:

(i) **Local miss rate of L2** (1 mark)

The local miss rate of L2 is the fraction of accesses reaching L2 that miss in L2.

$$MR_{L2}^{\text{local}} = \frac{\text{L2 misses}}{\text{L1 misses}} = \frac{20}{40} = \boxed{50\%}$$

(ii) **Global miss rate of L2** (1 mark)

The global miss rate of L2 is the fraction of all CPU-to-memory references that miss in L2.

$$MR_{L2}^{\text{global}} = \frac{\text{L2 misses}}{\text{Total memory references}} = \frac{20}{1000} = \boxed{2\%}$$

(iii) **Local miss rate of L3** (1 mark)

The local miss rate of L3 is the fraction of accesses reaching L3 that miss in L3.

$$MR_{L3}^{\text{local}} = \frac{\text{L3 misses}}{\text{L2 misses}} = \frac{10}{20} = \boxed{50\%}$$

(iv) **Global miss rate of L3** (1 mark)

The global miss rate of L3 is the fraction of all CPU-to-memory references that miss in L3.

$$MR_{L3}^{\text{global}} = \frac{\text{L3 misses}}{\text{Total memory references}} = \frac{10}{1000} = \boxed{1\%}$$

(v) **Average Memory Access Time (AMAT)** (2 marks)

The AMAT for a three-level cache hierarchy is

$$AMAT = T_{L1} + MR_{L1}\Big(T_{L2} + MR_{L2}^{\text{local}}\big(T_{L3} + MR_{L3}^{\text{local}} T_{\text{mem}}\big)\Big).$$

Substituting the given values,

$$\begin{aligned} AMAT &= 1 + 0.04\,(10 + 0.5(100 + 0.5 \times 400)) \\ &= 1 + 0.04(10 + 0.5 \times 300) \\ &= 1 + 0.04(160) \\ &= 1 + 6.4. \end{aligned}$$

Therefore,

$$\boxed{AMAT = 7.4 \text{ clock cycles}}$$

(vi) **Effective CPI with memory stalls** (2 marks)

The ideal CPI is 1, and there are 1.25 memory references per instruction.

Since the ideal CPI already accounts for L1 hits, only the additional latency beyond the L1 hit time contributes to memory stalls.

Thus,

$$CPI_{\text{effective}} = CPI_{\text{ideal}} + \text{Memory references per instruction} \times (AMAT - T_{L1}).$$

Substituting,

$$\begin{aligned} CPI_{\text{effective}} &= 1 + 1.25(7.4 - 1) \\ &= 1 + 1.25 \times 6.4 \\ &= 1 + 8. \end{aligned}$$

Therefore,

$$\boxed{CPI_{\text{effective}} = 9}$$

No part marks

8. A research laboratory is developing a new computing system for scientific simulations. The laboratory currently uses a baseline processor X, consisting of a single core with area A. The engineering team is exploring alternative processor architectures to improve application performance.

   All new processor designs must fit within a fixed silicon die area of $16A$. Assume that the single-thread performance of a core scales with the square root of its area.

   The scientific workload consists of two components: a serial fraction S, which must execute sequentially, and a perfectly parallelizable fraction $P = 1 - S$, which can be distributed across multiple cores.

The speedup of a processor T is defined relative to the baseline processor X as

$$\text{Speedup}(T) = \frac{\text{Execution time on } X}{\text{Execution time on } T}.$$

Answer the following questions:

(a) The engineering team first considers processor Z, which consists of a single large core occupying the entire die area of $16A$. What speedup does Z achieve for a workload with $S = 1$? (1 mark)

(b) The team now evaluates the same processor Z using a completely parallelizable workload ($S = 0$). What speedup does Z achieve? (1 mark)

(c) The team proposes a heterogeneous processor consisting of one large core of area $N \times A$, with the remaining die area occupied by small cores, each of area A.

Assume that only the large core executes the serial portion of the workload, whereas the parallel portion can execute on all cores, including the large core.

Derive an expression for the speedup of this heterogeneous processor in terms of S and N. (3 marks)

(d) Finally, the team is allowed to optimize the heterogeneous processor for a specific scientific workload with $P = 0.8$.

The large core must have an area of $M^2 \times A$, where M is a natural number, and the remaining die area is filled with small cores of area A each.

Determine the value of M that maximizes the processor's speedup. (2 marks)

Solution:

(a) **Speedup of processor Z for $S = 1$**

The single-thread performance of a core is proportional to the square root of its area.

Processor X has area A, whereas processor Z has area $16A$.

Therefore, the performance improvement is

$$\text{Speedup}(Z) = \frac{\sqrt{16A}}{\sqrt{A}} = \sqrt{16} = 4.$$

Since $S = 1$, the entire workload is serial.

Hence,

$$\boxed{\text{Speedup}(Z) = 4}$$

(b) **Speedup of processor Z for $S = 0$**

When $S = 0$, the workload is completely parallelizable.

However, processor Z consists of only one large core of area $16A$. Therefore, it cannot exploit parallelism across multiple cores.

Its performance relative to processor X remains

$$\text{Speedup}(Z) = \frac{\sqrt{16A}}{\sqrt{A}} = 4.$$

Hence,

$$\boxed{\text{Speedup}(Z) = 4}$$

(c) **Speedup of the heterogeneous processor**

The total available die area is $16A$.

The heterogeneous processor contains:

- One large core of area NA.
- $(16 - N)$ small cores, each of area A.

Since single-thread performance scales with the square root of core area, the performance of the large core relative to processor X is

$$\frac{\sqrt{NA}}{\sqrt{A}} = \sqrt{N}.$$

Each small core has performance 1 relative to X.

Serial portion:

Only the large core executes the serial fraction S.

Therefore, the normalized execution time for the serial portion is

$$T_{\text{serial}} = \frac{S}{\sqrt{N}}.$$

Parallel portion:

All cores, including the large core, execute the parallel fraction $P = 1 - S$.

The aggregate performance of the processor is

$$P_{\text{total}} = \sqrt{N} + (16 - N).$$

Therefore, the normalized execution time for the parallel portion is

$$T_{\text{parallel}} = \frac{1 - S}{\sqrt{N} + 16 - N}.$$

The total normalized execution time is

$$T_{\text{total}} = \frac{S}{\sqrt{N}} + \frac{1-S}{\sqrt{N}+16-N}.$$

Since the execution time on processor X is normalized to 1, the speedup is

$$\boxed{\text{Speedup}(S,N) = \frac{1}{\frac{S}{\sqrt{N}} + \frac{1-S}{\sqrt{N}+16-N}}}$$

where $1 \leq N \leq 16$, and $16-N$ must be a nonnegative integer to accommodate whole small cores.

(d) **Optimal heterogeneous processor for** $P = 0.8$

Given

$$P = 0.8, \qquad S = 1 - P = 0.2.$$

The large core has area M^2A.

Therefore,

$$N = M^2, \qquad \sqrt{N} = M.$$

Substituting into the speedup equation from part (c),

$$\text{Speedup}(M) = \frac{1}{\frac{0.2}{M} + \frac{0.8}{M+16-M^2}}.$$

Since the total die area is $16A$,

$$M^2A \leq 16A \implies M^2 \leq 16.$$

As M is a natural number,

$$M \in \{1, 2, 3, 4\}.$$

We evaluate the speedup for each possible value.

Case 1: $M = 1$

$$\begin{aligned}\text{Speedup}(1) &= \frac{1}{\frac{0.2}{1} + \frac{0.8}{1+16-1}} \\ &= \frac{1}{0.2 + 0.05} \\ &= 4.\end{aligned}$$

Case 2: $M = 2$

$$\begin{aligned}\text{Speedup}(2) &= \frac{1}{\frac{0.2}{2} + \frac{0.8}{2+16-4}} \\ &= \frac{1}{0.1 + \frac{0.8}{14}} \\ &\approx 6.36.\end{aligned}$$

Case 3: $M = 3$

$$\begin{aligned}\text{Speedup}(3) &= \frac{1}{\frac{0.2}{3} + \frac{0.8}{3+16-9}} \\ &= \frac{1}{\frac{0.2}{3} + 0.08} \\ &\approx 6.82.\end{aligned}$$

Case 4: $M = 4$

$$\begin{aligned}\text{Speedup}(4) &= \frac{1}{\frac{0.2}{4} + \frac{0.8}{4+16-16}} \\ &= \frac{1}{0.05 + 0.2} \\ &= 4.\end{aligned}$$

The results are summarized below:

M	Large Core Area	Small Cores	Speedup
1	A	15	4.00
2	$4A$	12	6.36
3	$9A$	7	6.82
4	$16A$	0	4.00

The maximum speedup occurs for $M = 3$.

Therefore,

$$\boxed{M = 3}$$

The optimal processor contains one large core of area $9A$ and seven small cores of area A each, achieving a maximum speedup of approximately

$$\boxed{\text{Maximum Speedup} \approx 6.82}.$$

(a),(b) no part marks. (c) give 1, if the expression of execution time for the serial part is correct. Give 3 only when the entire expression is correct. (d) Give 2 if the approach is correct. Give full marks only if the total answer is correct.

9. Design a synchronous sequential circuit that generates the following repeating sequence at its two-bit output Y_1Y_0:

$$00 \to 01 \to 00 \to 10 \to 00 \to 11 \to 00 \to \cdots$$

The circuit must produce one value from the sequence per clock cycle, advancing to the next value at every rising clock edge. After producing 11, it must return to the first occurrence of 00 and repeat the sequence indefinitely.

All flip-flops are positive-edge-triggered and driven by a common clock. You may use any internal state encoding and combinational logic to generate the two-bit output from the flip-flop outputs. The output Y_1Y_0 need not be identical to any two internal state bits.

Assume that the circuit is initialized to the internal state corresponding to the first occurrence of 00. The output must be correct immediately after initialization and remain stable between clock edges, ignoring propagation delays.

(a) Determine the minimum number of flip-flops required to implement the circuit. Justify your answer. (3 marks)

(b) Construct the circuit using the minimum number of D flip-flops. Provide the internal state encoding, state-transition table, next-state equations, and combinational output equations. (7 marks)

Note: Unused internal states may be treated as don't-care conditions when deriving the logic equations. Recovery from unused states is not required.

Solution:

(a) **Minimum number of flip-flops (3 marks)**

The required output sequence is

$$00 \to 01 \to 00 \to 10 \to 00 \to 11 \to 00 \to \cdots$$

Although the output has only four distinct values, the sequence has six distinguishable positions.

In particular, the three occurrences of 00 must correspond to different internal states, since their subsequent outputs are 01, 10, and 11, respectively.

Thus, the circuit requires at least six distinct internal states.

With n flip-flops, at most 2^n states can be represented. Therefore,

$$2^n \geq 6$$

which gives

$$n \geq \lceil \log_2 6 \rceil = 3.$$

Hence, the minimum number of flip-flops is

$$\boxed{n_{\min} = 3}.$$

(b) **Circuit construction (7 marks)**

We use three D flip-flops with outputs Q_2, Q_1, Q_0.

The internal states follow a MOD-6 counting sequence:

$$000 \to 001 \to 010 \to 011 \to 100 \to 101 \to 000.$$

The two-bit output Y_1Y_0 is generated using combinational logic.

State-transition table

Present State $Q_2Q_1Q_0$	Next State $Q_2^+Q_1^+Q_0^+$	Output Y_1Y_0
000	001	00
001	010	01
010	011	00
011	100	10
100	101	00
101	000	11
110	XXX	XX
111	XXX	XX

The unused states 110 and 111 are treated as don't-care conditions.

Next-state equations

For D flip-flops,

$$D_i = Q_i^+.$$

From the state-transition table, one possible set of next-state equations is

$$\boxed{D_0 = \overline{Q_0}}$$

$$\boxed{D_1 = \overline{Q_2}(Q_1 \oplus Q_0)}$$

$$\boxed{D_2 = Q_1Q_0 + Q_2\overline{Q_0}}$$

These equations implement the required six-state counting sequence.

Output equations

From the state-transition table, $Y_1 = 1$ in states 011 and 101.

Therefore,

$$\begin{aligned} Y_1 &= \overline{Q_2}Q_1Q_0 + Q_2\overline{Q_1}Q_0 \\ &= Q_0(Q_1 \oplus Q_2). \end{aligned}$$

Hence,

$$\boxed{Y_1 = Q_0(Q_1 \oplus Q_2)}$$

Similarly, $Y_0 = 1$ in states 001 and 101.

Thus,

$$\begin{aligned} Y_0 &= \overline{Q_2 Q_1} Q_0 + Q_2 \overline{Q_1} Q_0 \\ &= \overline{Q_1} Q_0. \end{aligned}$$

Therefore,

$$\boxed{Y_0 = \overline{Q_1} Q_0}$$

Verification

Starting from the initialized state 000, the circuit produces

Internal State	000	001	010	011	100	101
$Y_1 Y_0$	00	01	00	10	00	11

After state 101, the circuit returns to 000, and the output sequence repeats.

Thus, the circuit generates exactly the required sequence using three D flip-flops.

$$\boxed{\text{Minimum flip-flops required} = 3}$$

(a)There can be two answers. If someone is only using DFFs without any output logic, the answer can be 4. Else, with output logic the answer will be 3. Give marks for both cases, only if correctly articulated. b) Give 2 marks for each of the state transition table, next state equations, and output equations. Full marks only if everything is correct.

10. Design an 8-to-1 multiplexer using exactly four 2-to-1 multiplexers and one 4-to-1 multiplexer. No additional logic gates or multiplexers are allowed.

    The 8-to-1 multiplexer has eight single-bit data inputs $I_0, I_1, \ldots, I_7$, three select inputs S_2, S_1, S_0, and one output Y.

    A 2-to-1 multiplexer selects input 0 when its select input is 0 and input 1 when it is 1. Similarly, a 4-to-1 multiplexer selects input j when its two-bit select value is j, where $j \in \{0, 1, 2, 3\}$.

    Draw the complete circuit diagram, clearly labeling all data inputs, select inputs, intermediate signals, and the final output. (5 marks)

Solution:

The 8-to-1 multiplexer can be constructed in two stages using four 2-to-1 multiplexers and one 4-to-1 multiplexer.

Stage 1: Four 2-to-1 Multiplexers

Connect the eight data inputs in pairs to the four 2-to-1 multiplexers. Use S_0 as the common select input for all four multiplexers.

The outputs of the four multiplexers are

$$\begin{aligned} M_0 &= \overline{S_0} I_0 + S_0 I_1, \\ M_1 &= \overline{S_0} I_2 + S_0 I_3, \\ M_2 &= \overline{S_0} I_4 + S_0 I_5, \\ M_3 &= \overline{S_0} I_6 + S_0 I_7. \end{aligned}$$

Stage 2: One 4-to-1 Multiplexer

Connect M_0, M_1, M_2, M_3 to data inputs D_0, D_1, D_2, D_3, respectively, of the 4-to-1 multiplexer.

Use S_2 and S_1 as the most significant and least significant select inputs, respectively.

The final output is

$$\begin{aligned} Y = &\overline{S_2}\,\overline{S_1} M_0 + \overline{S_2} S_1 M_1 \\ &+ S_2 \overline{S_1} M_2 + S_2 S_1 M_3. \end{aligned}$$

1 marks for getting the first stage/second stage correct. Full marks only if completely correct.