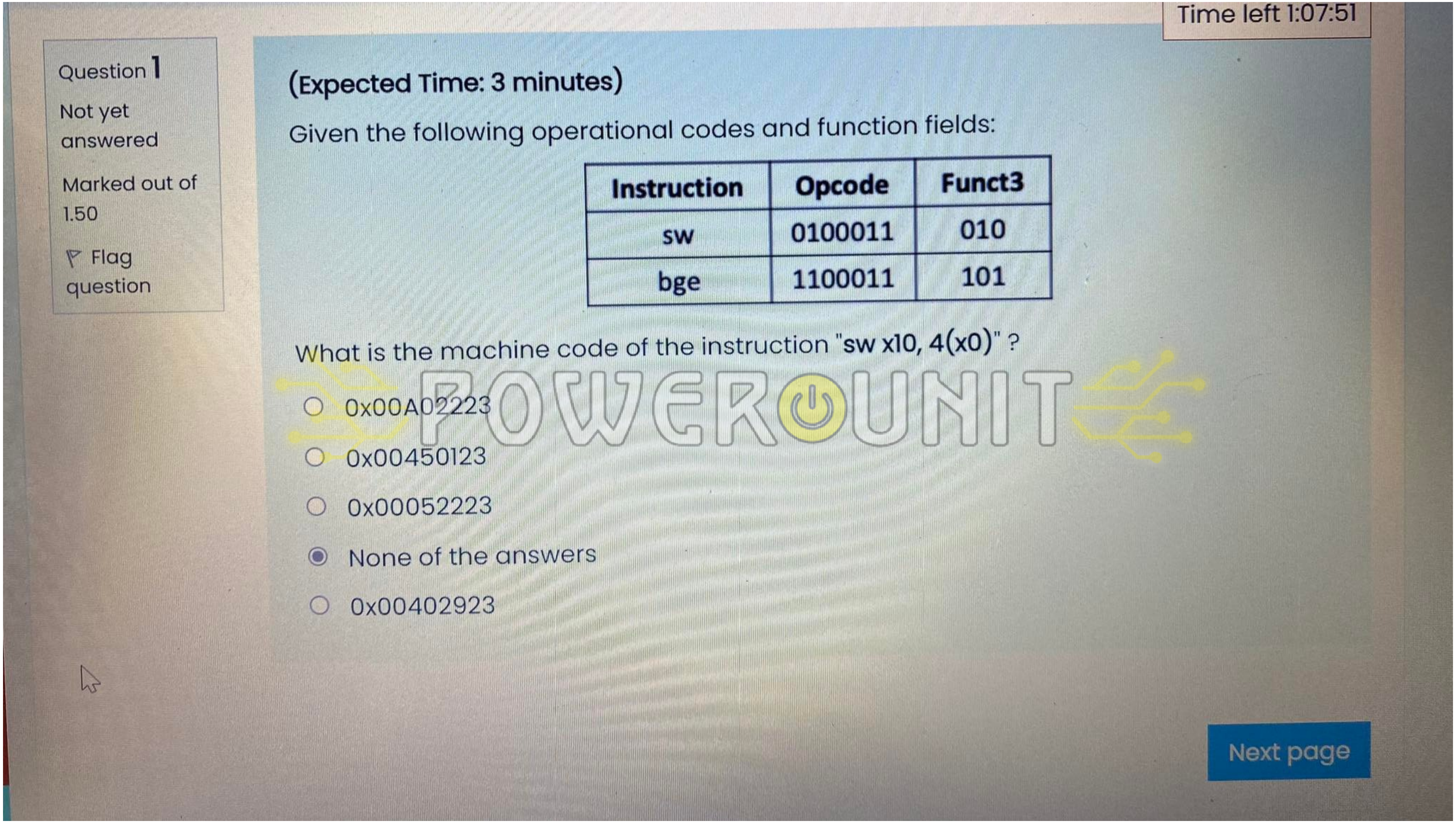


Time left 1:07:51

Question 1

Not yet answered

Marked out of 1.50

Flag question

(Expected Time: 3 minutes)

Given the following operational codes and function fields:

Instruction	Opcode	Funct3
sw	0100011	010
bge	1100011	101

What is the machine code of the instruction "**sw x10, 4(x0)**" ?

○ 0x00A02223

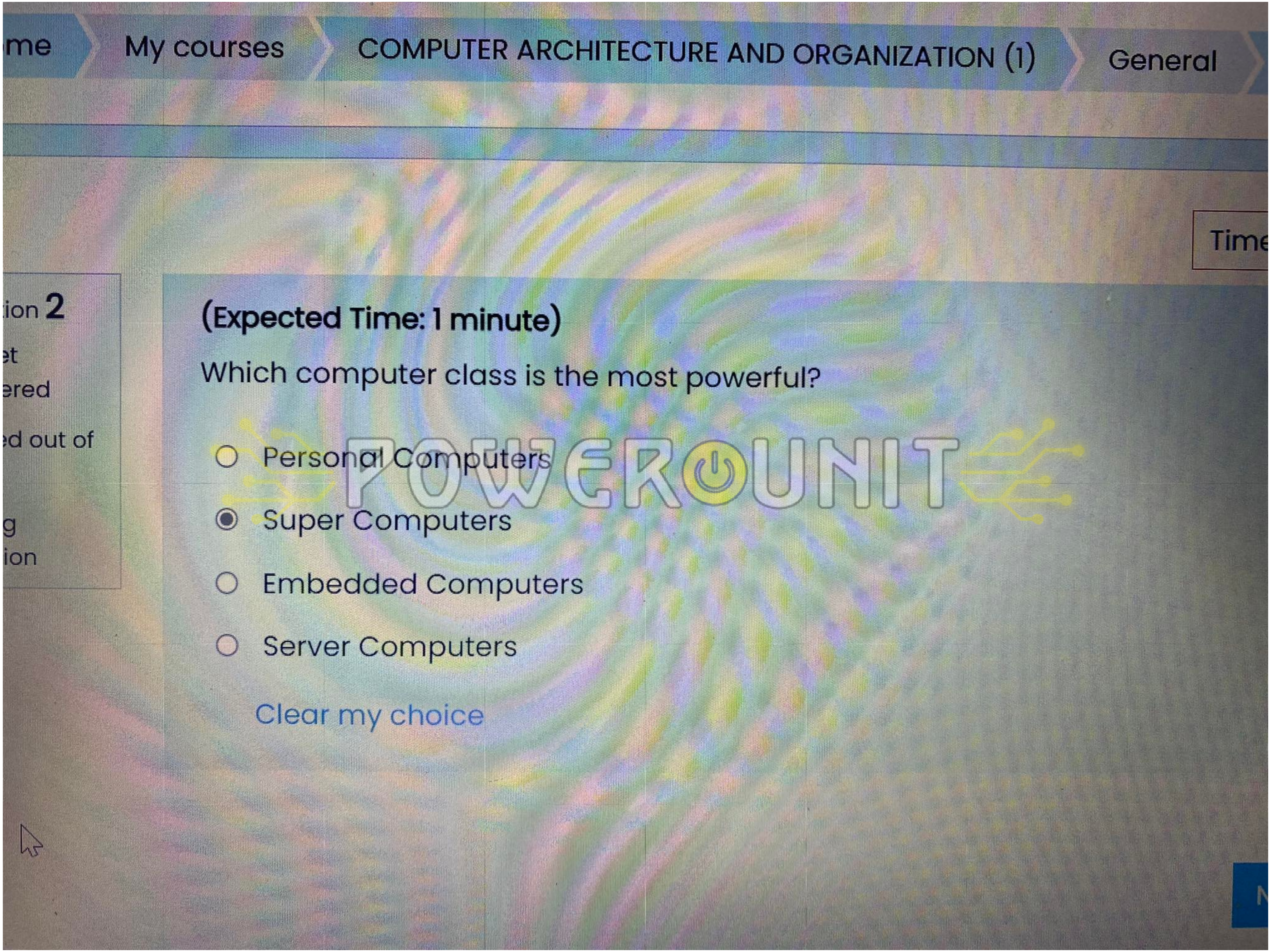
○ 0x00450123

○ 0x00052223

● None of the answers

○ 0x00402923

Next page



(Expected Time: 1 minute)

Which computer class is the most powerful?

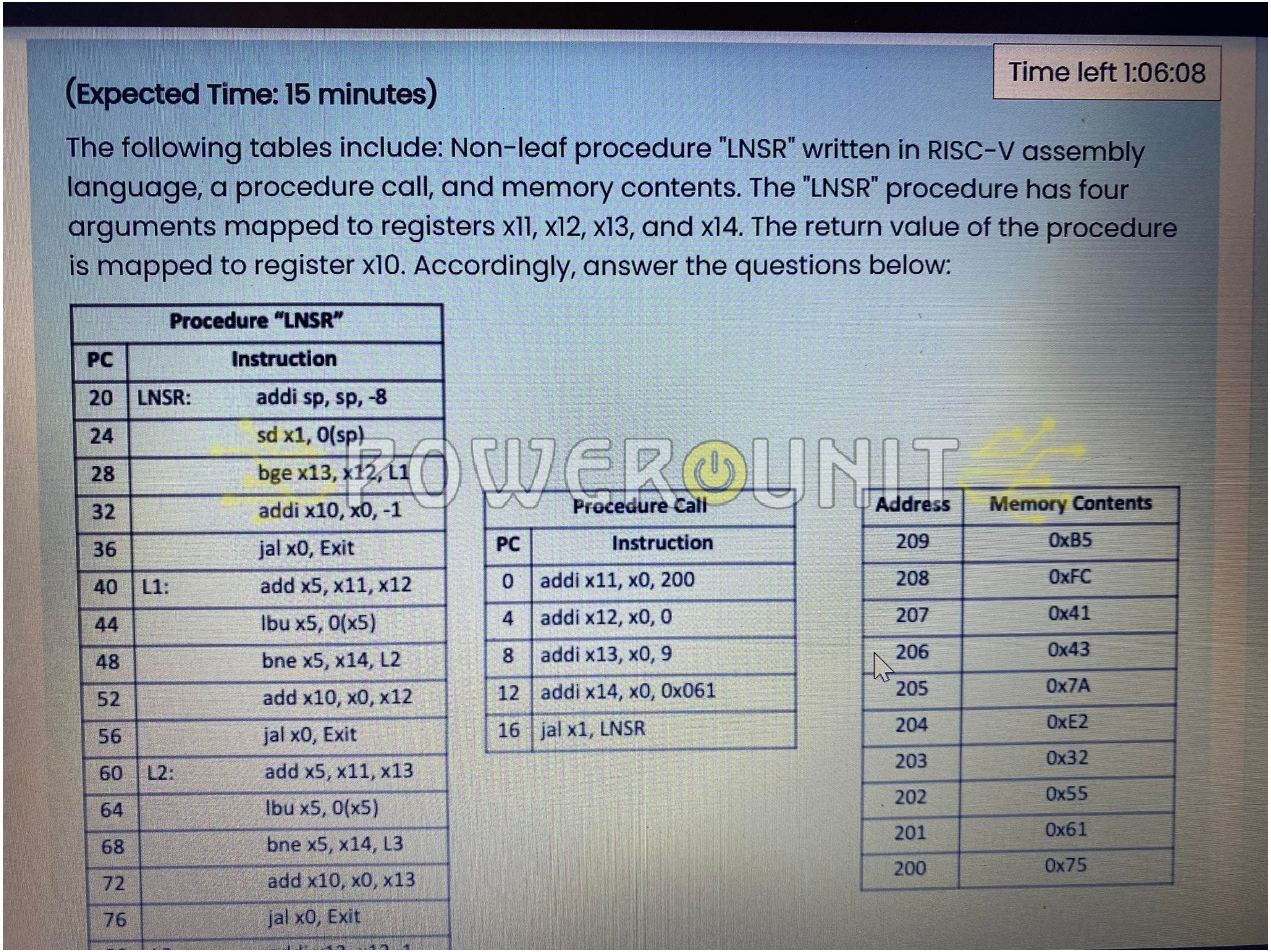
○ Personal Computers

● Super Computers

○ Embedded Computers

○ Server Computers

Clear my choice



Time left 1:06:08

(Expected Time: 15 minutes)

The following tables include: Non-leaf procedure "LNSR" written in RISC-V assembly language, a procedure call, and memory contents. The "LNSR" procedure has four arguments mapped to registers x11, x12, x13, and x14. The return value of the procedure is mapped to register x10. Accordingly, answer the questions below:

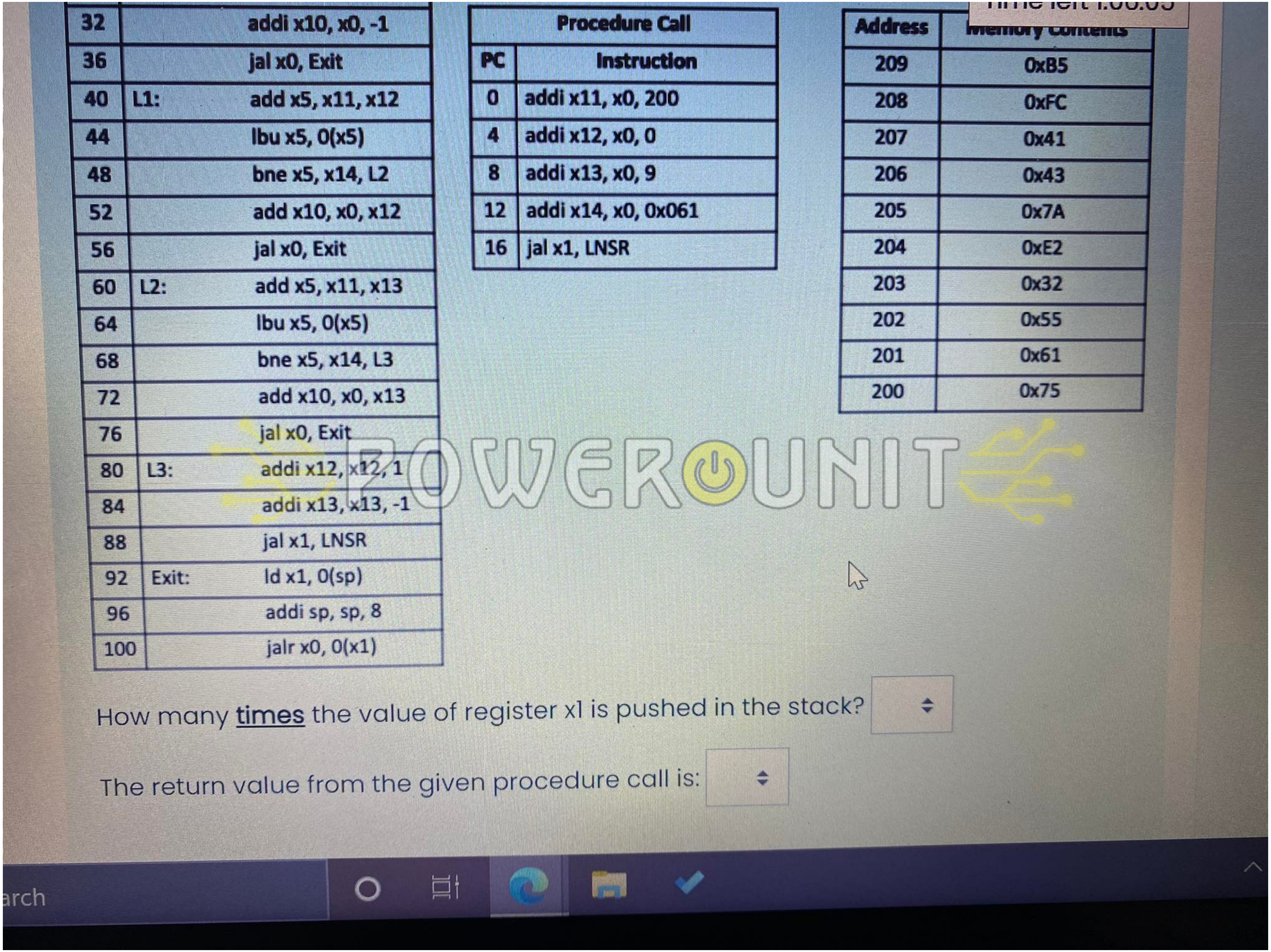
Procedure "LNSR"

PC	Instruction	
20	LNSR:	addi sp, sp, -8
24		sd x1, 0(sp)
28		bge x13, x12, L1
32		addi x10, x0, -1
36		jal x0, Exit
40	L1:	add x5, x11, x12
44		lbu x5, 0(x5)
48		bne x5, x14, L2
52		add x10, x0, x12
56		jal x0, Exit
60	L2:	add x5, x11, x13
64		lbu x5, 0(x5)
68		bne x5, x14, L3
72		add x10, x0, x13
76		jal x0, Exit

Procedure Call

PC	Instruction
0	addi x11, x0, 200
4	addi x12, x0, 0
8	addi x13, x0, 9
12	addi x14, x0, 0x061
16	jal x1, LNSR

Address	Memory Contents
209	0xB5
208	0xFC
207	0x41
206	0x43
205	0x7A
204	0xE2
203	0x32
202	0x55
201	0x61
200	0x75



Time left 1:00:05

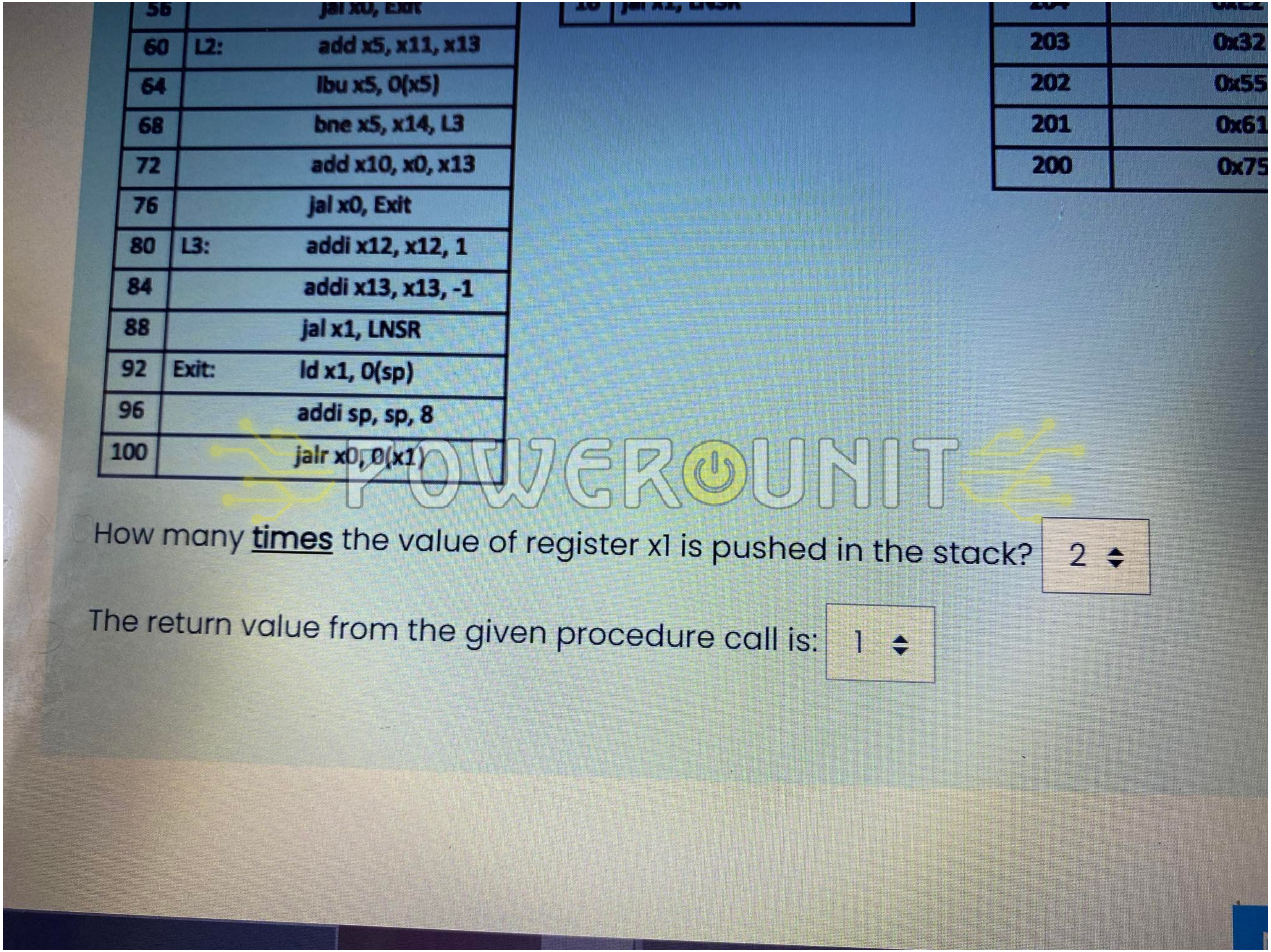
32		addi x10, x0, -1
36		jal x0, Exit
40	L1:	add x5, x11, x12
44		lbu x5, 0(x5)
48		bne x5, x14, L2
52		add x10, x0, x12
56		jal x0, Exit
60	L2:	add x5, x11, x13
64		lbu x5, 0(x5)
68		bne x5, x14, L3
72		add x10, x0, x13
76		jal x0, Exit
80	L3:	addi x12, x12, 1
84		addi x13, x13, -1
88		jal x1, LNSR
92	Exit:	ld x1, 0(sp)
96		addi sp, sp, 8
100		jalr x0, 0(x1)

Procedure Call	
PC	Instruction
0	addi x11, x0, 200
4	addi x12, x0, 0
8	addi x13, x0, 9
12	addi x14, x0, 0x061
16	jal x1, LNSR

Address	Memory Contents
209	0xB5
208	0xFC
207	0x41
206	0x43
205	0x7A
204	0xE2
203	0x32
202	0x55
201	0x61
200	0x75

How many times the value of register x1 is pushed in the stack? ⬍

The return value from the given procedure call is: ⬍



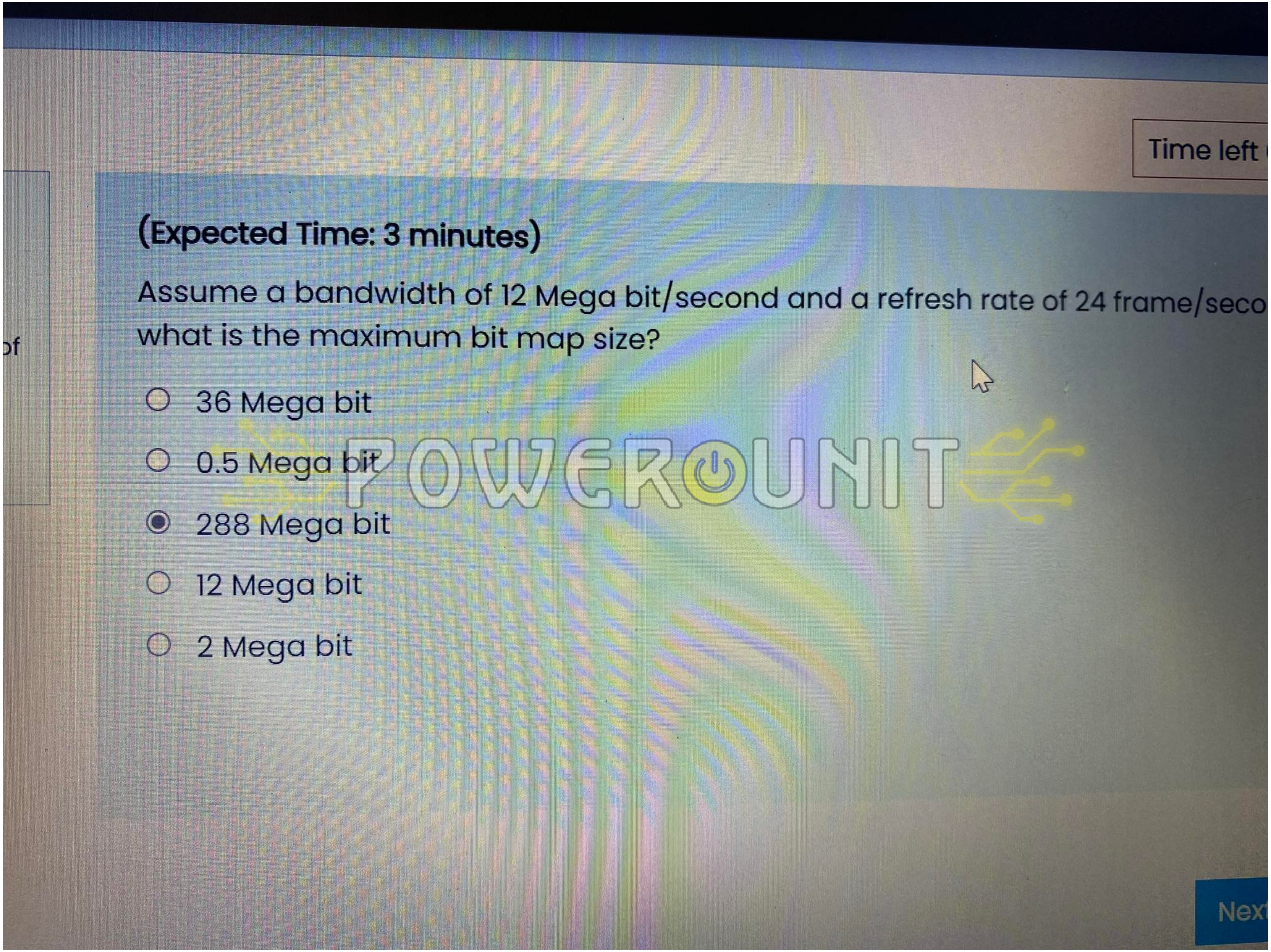
56		jal x0, Exit
60	L2:	add x5, x11, x13
64		lbu x5, 0(x5)
68		bne x5, x14, L3
72		add x10, x0, x13
76		jal x0, Exit
80	L3:	addi x12, x12, 1
84		addi x13, x13, -1
88		jal x1, LNSR
92	Exit:	ld x1, 0(sp)
96		addi sp, sp, 8
100		jalr x0, 0(x1)

16	jal x1, LNSR

204	0xE2
203	0x32
202	0x55
201	0x61
200	0x75

How many **times** the value of register x1 is pushed in the stack? 2 ⬍

The return value from the given procedure call is: 1 ⬍



of

(Expected Time: 3 minutes)

Assume a bandwidth of 12 Mega bit/second and a refresh rate of 24 frame/seco
what is the maximum bit map size?

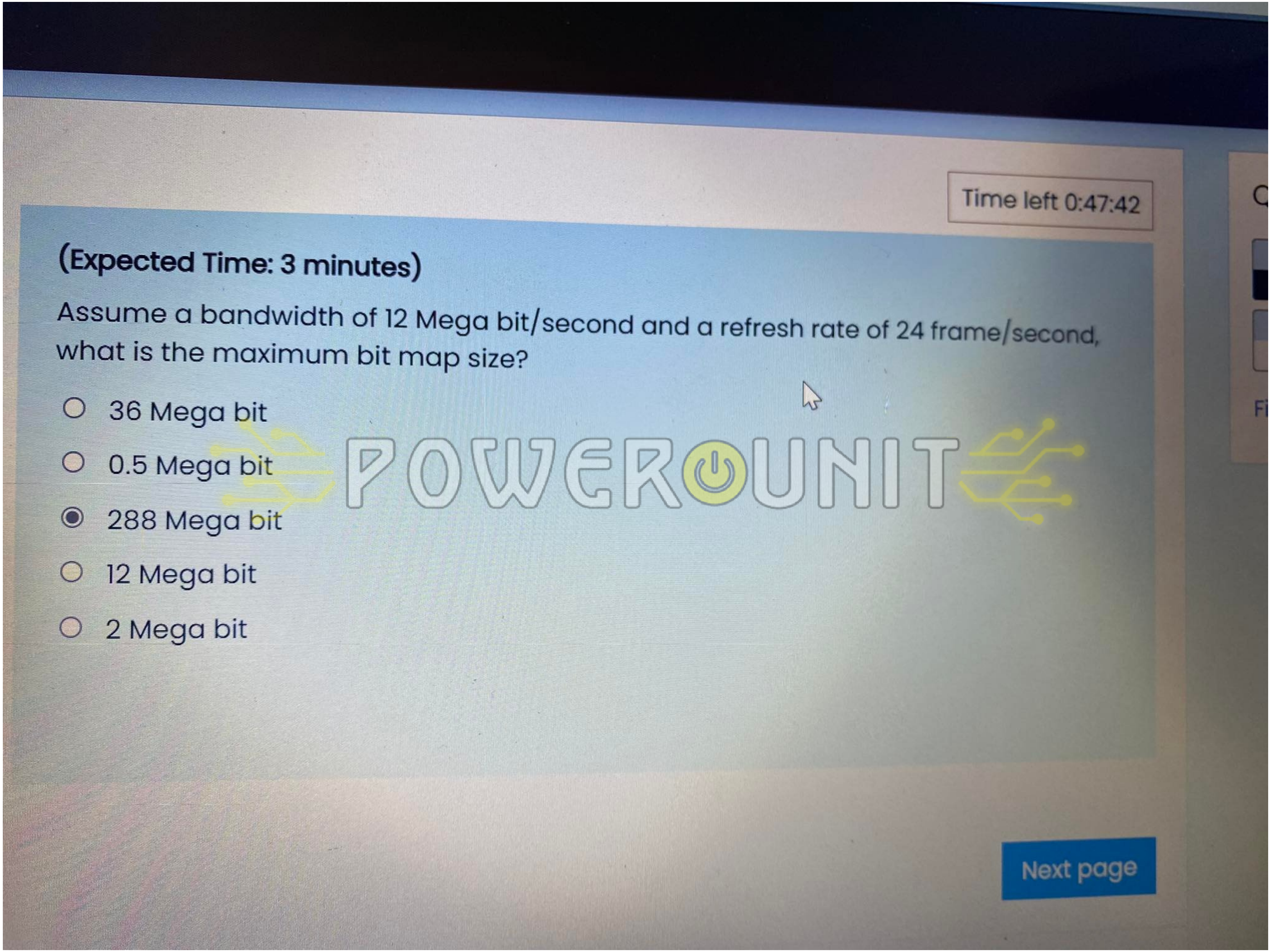
○ 36 Mega bit

○ 0.5 Mega bit

● 288 Mega bit

○ 12 Mega bit

○ 2 Mega bit



Time left 0:47:42

(Expected Time: 3 minutes)

Assume a bandwidth of 12 Mega bit/second and a refresh rate of 24 frame/second, what is the maximum bit map size?

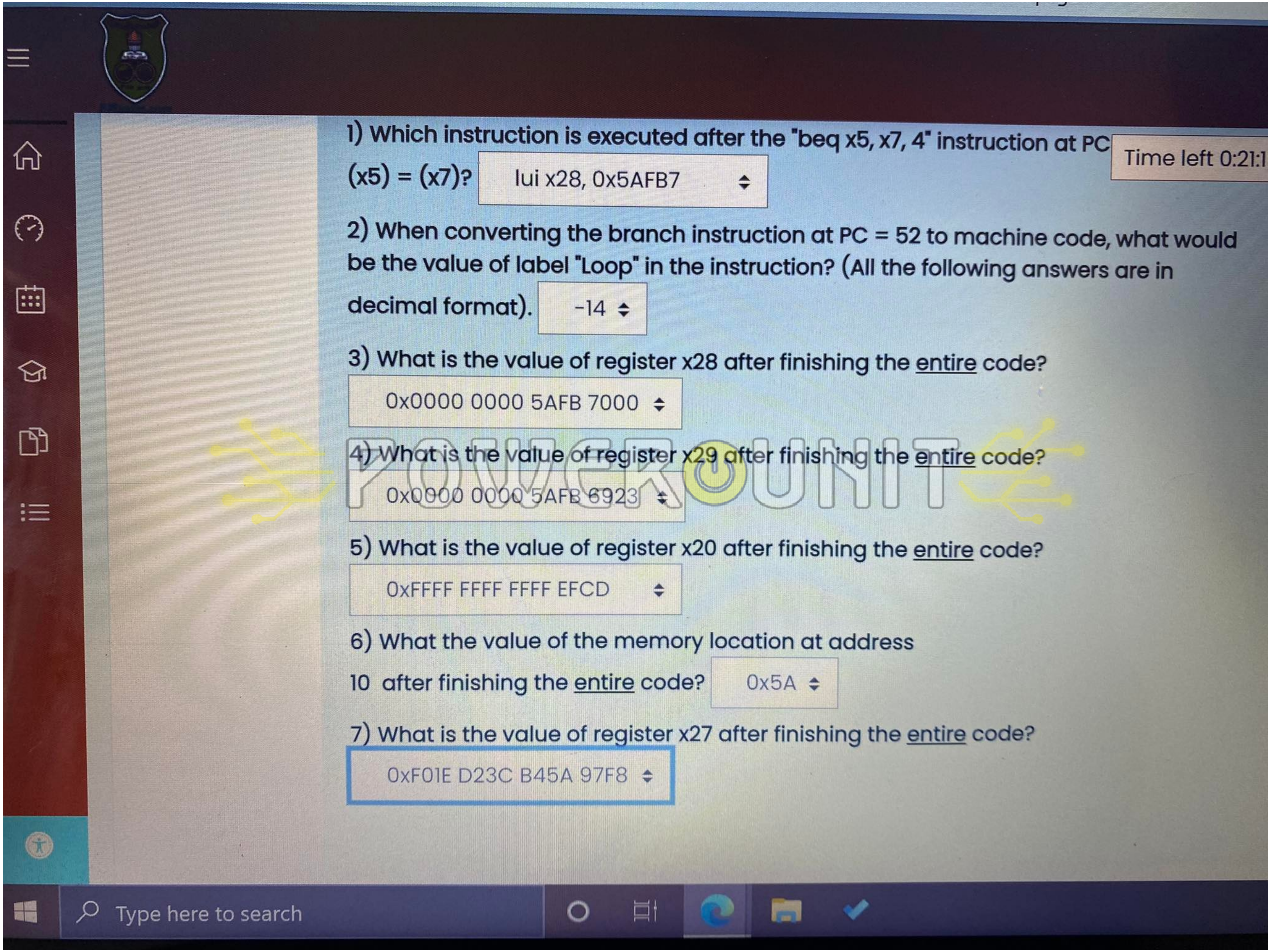
○ 36 Mega bit

○ 0.5 Mega bit

● 288 Mega bit

○ 12 Mega bit

○ 2 Mega bit



1) Which instruction is executed after the "beq x5, x7, 4" instruction at PC (x5) = (x7)? lui x28, 0x5AFB7 ⬍

Time left 0:21:1

2) When converting the branch instruction at PC = 52 to machine code, what would be the value of label "Loop" in the instruction? (All the following answers are in decimal format). -14 ⬍

3) What is the value of register x28 after finishing the entire code?

0x0000 0000 5AFB 7000 ⬍

4) What is the value of register x29 after finishing the entire code?

0x0000 0000 5AFB 6923 ⬍

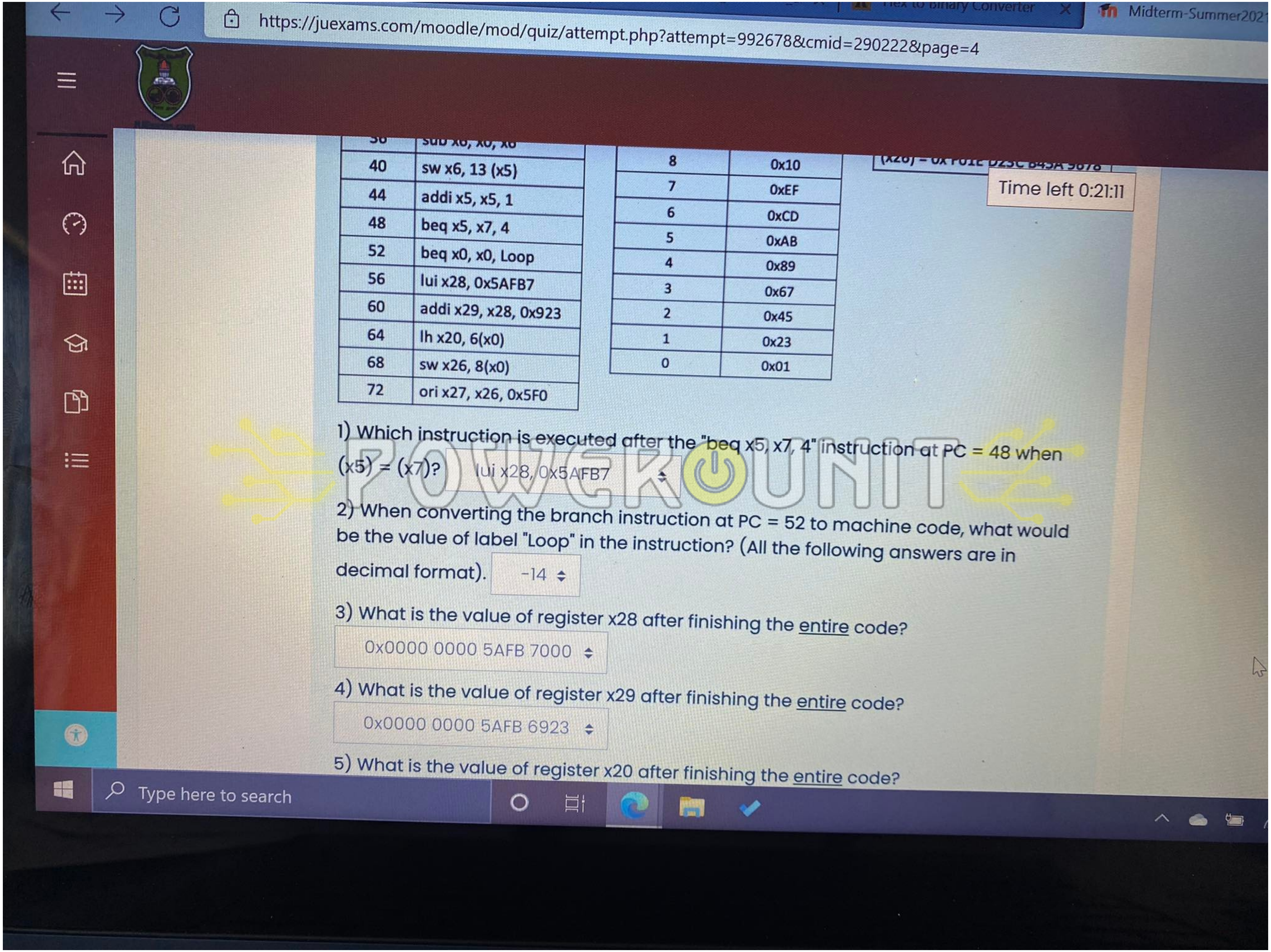
5) What is the value of register x20 after finishing the entire code?

0xFFFF FFFF FFFF EFCD ⬍

6) What the value of the memory location at address 10 after finishing the entire code? 0x5A ⬍

7) What is the value of register x27 after finishing the entire code?

0xF01E D23C B45A 97F8 ⬍



https://juexams.com/moodle/mod/quiz/attempt.php?attempt=992678&cmid=290222&page=4

36	sub x6, x0, x6
40	sw x6, 13 (x5)
44	addi x5, x5, 1
48	beq x5, x7, 4
52	beq x0, x0, Loop
56	lui x28, 0x5AFB7
60	addi x29, x28, 0x923
64	lh x20, 6(x0)
68	sw x26, 8(x0)
72	ori x27, x26, 0x5F0

8	0x10
7	0xEF
6	0xCD
5	0xAB
4	0x89
3	0x67
2	0x45
1	0x23
0	0x01

(x26) = 0x F01E D23C B45A 5678

Time left 0:21:11

1) Which instruction is executed after the "beq x5, x7, 4" instruction at PC = 48 when (x5) = (x7)? lui x28, 0x5AFB7 ⬍

2) When converting the branch instruction at PC = 52 to machine code, what would be the value of label "Loop" in the instruction? (All the following answers are in decimal format). -14 ⬍

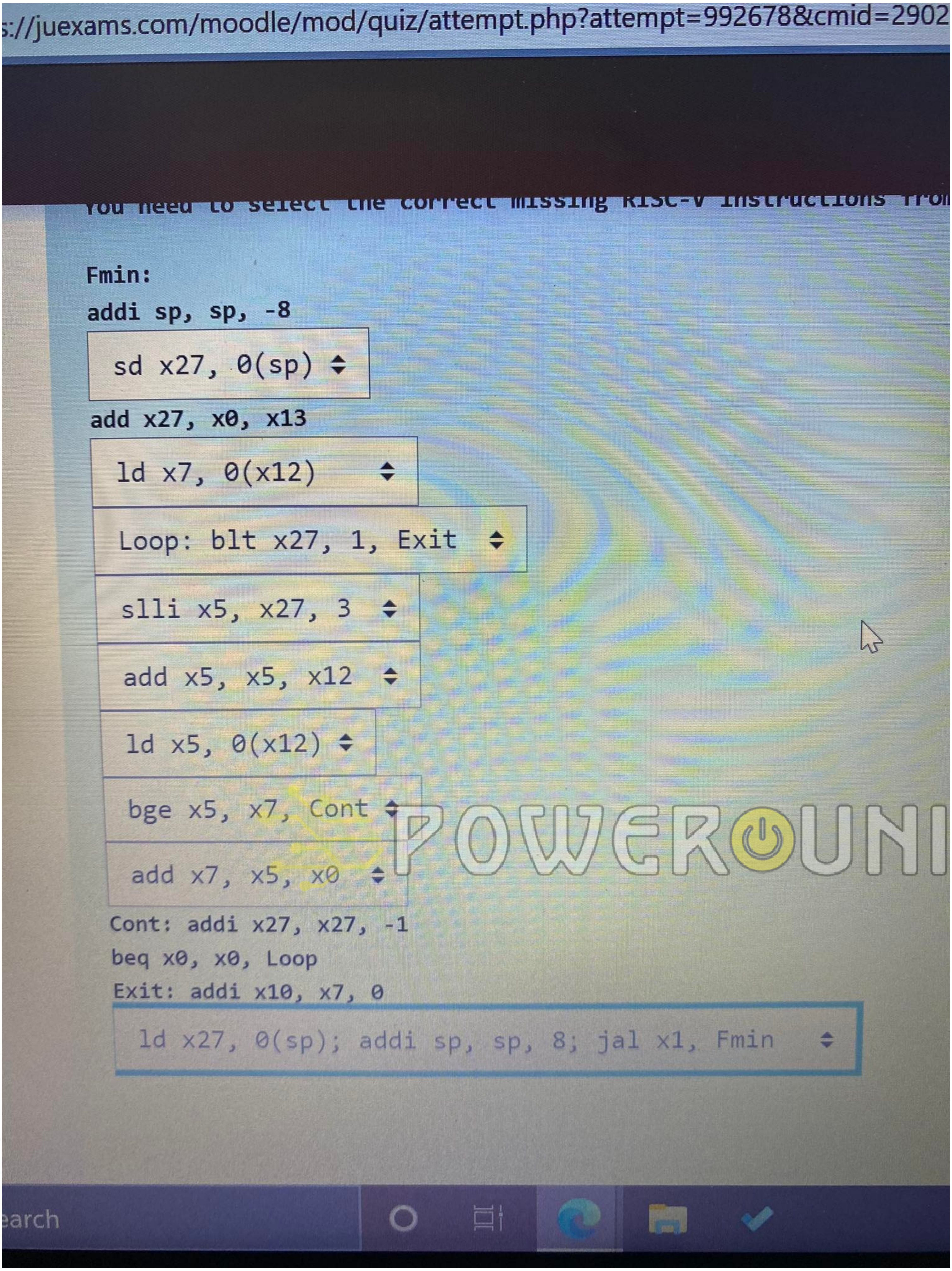
3) What is the value of register x28 after finishing the entire code?

0x0000 0000 5AFB 7000 ⬍

4) What is the value of register x29 after finishing the entire code?

0x0000 0000 5AFB 6923 ⬍

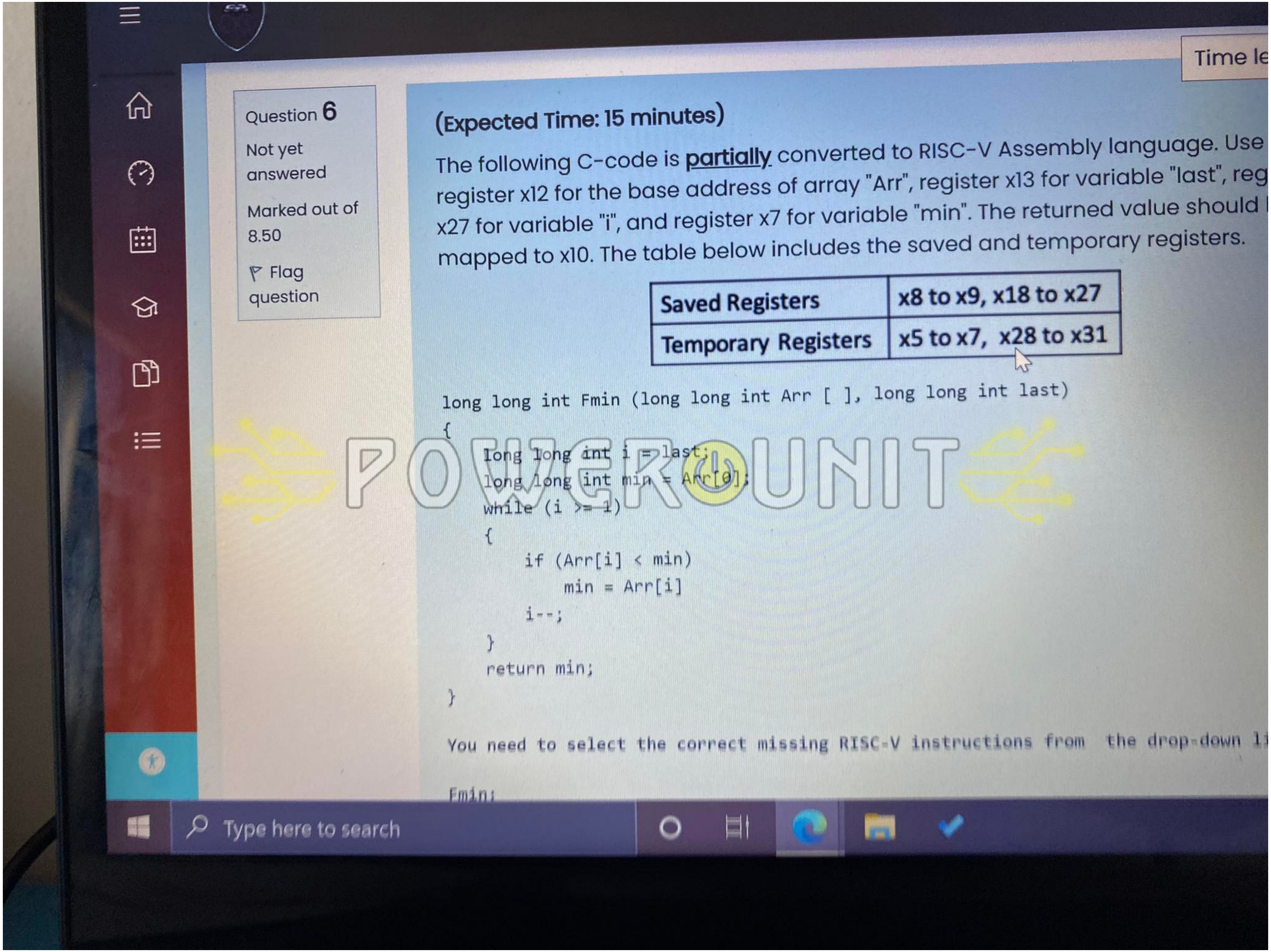
5) What is the value of register x20 after finishing the entire code?



s://juexams.com/moodle/mod/quiz/attempt.php?attempt=992678&cmid=2902

You need to select the correct missing RISC-V instructions from

```
Fmin:
addi sp, sp, -8
sd x27, 0(sp)
add x27, x0, x13
ld x7, 0(x12)
Loop: blt x27, 1, Exit
slli x5, x27, 3
add x5, x5, x12
ld x5, 0(x12)
bge x5, x7, Cont
add x7, x5, x0
Cont: addi x27, x27, -1
beq x0, x0, Loop
Exit: addi x10, x7, 0
ld x27, 0(sp); addi sp, sp, 8; jal x1, Fmin
```



Time le

Question 6

Not yet answered

Marked out of 8.50

Flag question

(Expected Time: 15 minutes)

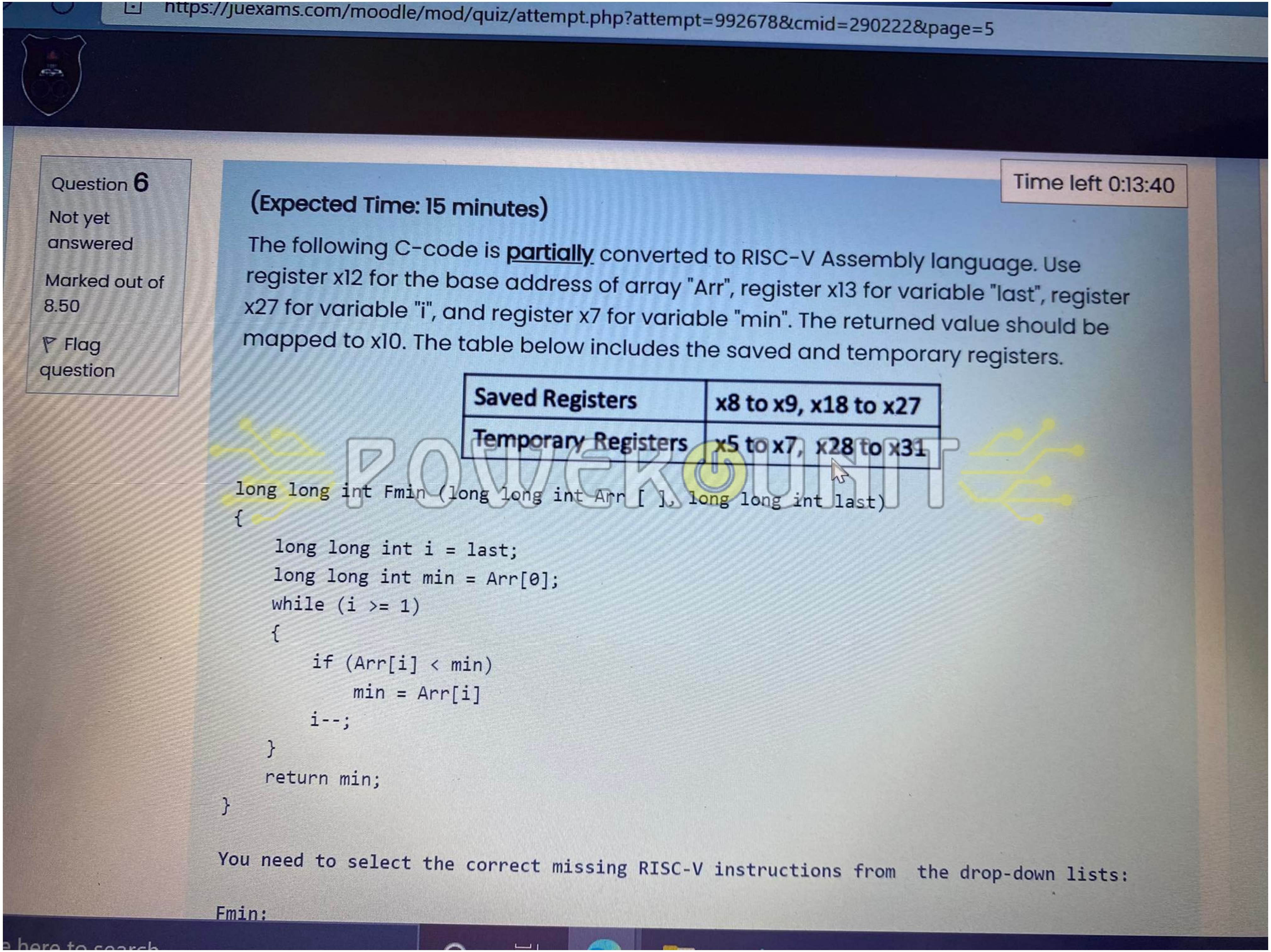
The following C-code is **partially** converted to RISC-V Assembly language. Use register x12 for the base address of array "Arr", register x13 for variable "last", reg x27 for variable "i", and register x7 for variable "min". The returned value should mapped to x10. The table below includes the saved and temporary registers.

Saved Registers	x8 to x9, x18 to x27
Temporary Registers	x5 to x7, x28 to x31

```
long long int Fmin (long long int Arr [ ], long long int last)
{
    long long int i = last;
    long long int min = Arr[0];
    while (i >= 1)
    {
        if (Arr[i] < min)
            min = Arr[i]
        i--;
    }
    return min;
}
```

You need to select the correct missing RISC-V instructions from the drop-down li

```
Fmin:
```



https://juexams.com/moodle/mod/quiz/attempt.php?attempt=992678&cmid=290222&page=5

Question **6**

Not yet answered

Marked out of 8.50

Flag question

Time left 0:13:40

(Expected Time: 15 minutes)

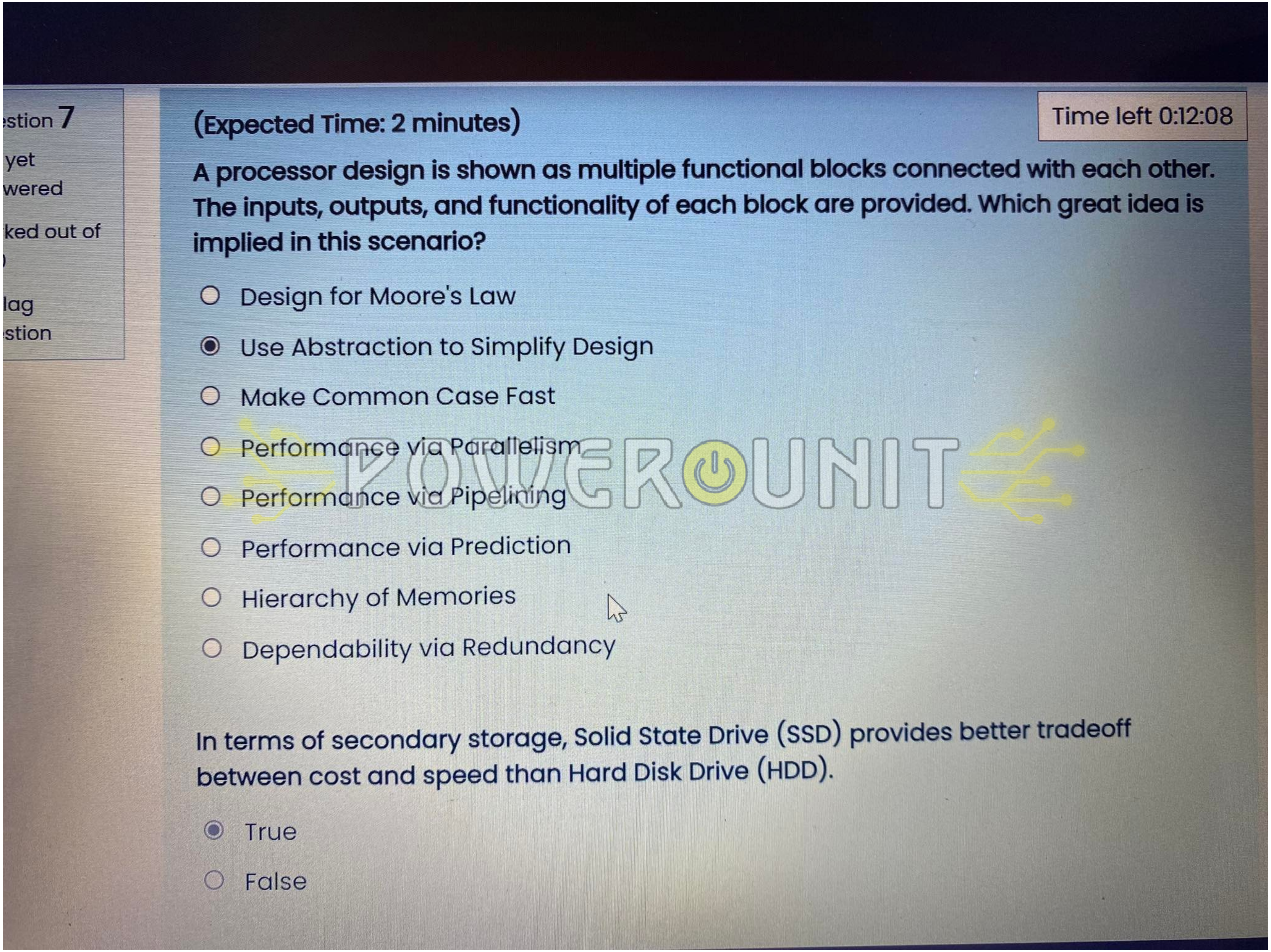
The following C-code is **partially** converted to RISC-V Assembly language. Use register x12 for the base address of array "Arr", register x13 for variable "last", register x27 for variable "i", and register x7 for variable "min". The returned value should be mapped to x10. The table below includes the saved and temporary registers.

Saved Registers	**x8 to x9, x18 to x27**
Temporary Registers	**x5 to x7, x28 to x31**

```
long long int Fmin (long long int Arr [ ], long long int last)
{
    long long int i = last;
    long long int min = Arr[0];
    while (i >= 1)
    {
        if (Arr[i] < min)
            min = Arr[i]
        i--;
    }
    return min;
}
```

You need to select the correct missing RISC-V instructions from the drop-down lists:

Fmin:



Time left 0:12:08

(Expected Time: 2 minutes)

A processor design is shown as multiple functional blocks connected with each other. The inputs, outputs, and functionality of each block are provided. Which great idea is implied in this scenario?

○ Design for Moore's Law

● Use Abstraction to Simplify Design

○ Make Common Case Fast

○ Performance via Parallelism

○ Performance via Pipelining

○ Performance via Prediction

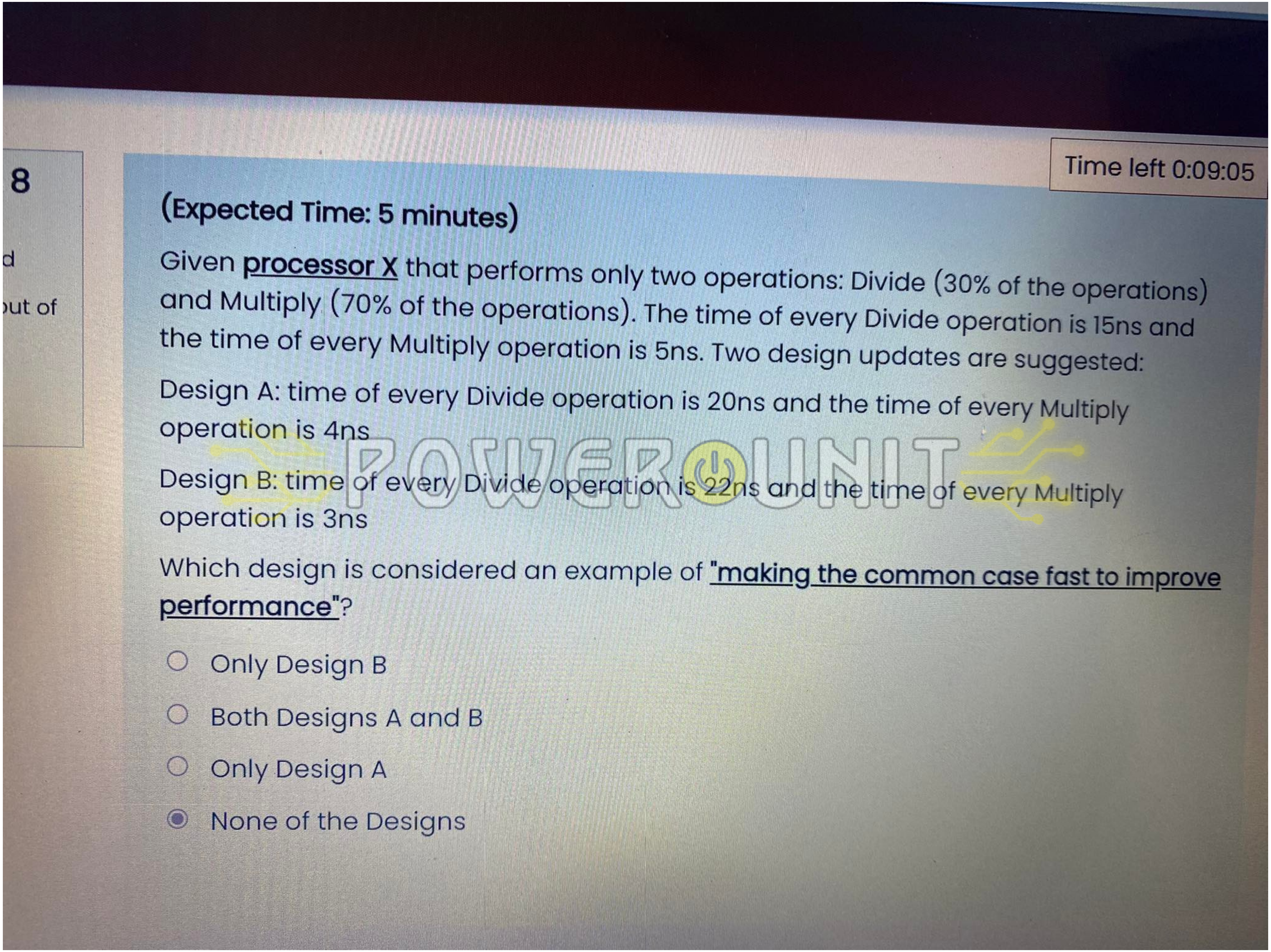
○ Hierarchy of Memories

○ Dependability via Redundancy

In terms of secondary storage, Solid State Drive (SSD) provides better tradeoff between cost and speed than Hard Disk Drive (HDD).

● True

○ False



8

d

ut of

Time left 0:09:05

(Expected Time: 5 minutes)

Given **processor X** that performs only two operations: Divide (30% of the operations) and Multiply (70% of the operations). The time of every Divide operation is 15ns and the time of every Multiply operation is 5ns. Two design updates are suggested:

Design A: time of every Divide operation is 20ns and the time of every Multiply operation is 4ns

Design B: time of every Divide operation is 22ns and the time of every Multiply operation is 3ns

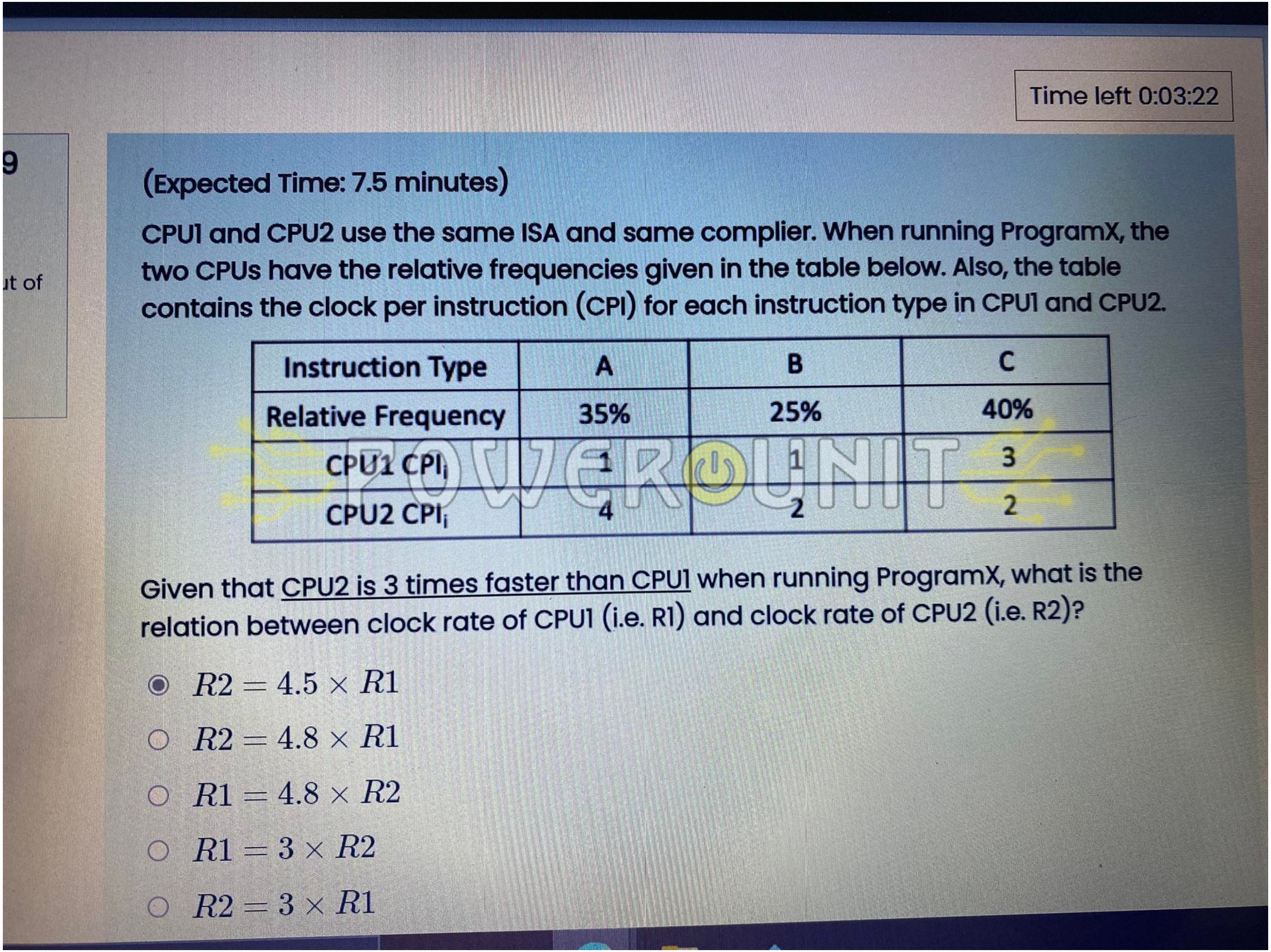
Which design is considered an example of **"making the common case fast to improve performance"**?

○ Only Design B

○ Both Designs A and B

○ Only Design A

● None of the Designs



Time left 0:03:22

9

ut of

(Expected Time: 7.5 minutes)

CPU1 and CPU2 use the same ISA and same complier. When running ProgramX, the two CPUs have the relative frequencies given in the table below. Also, the table contains the clock per instruction (CPI) for each instruction type in CPU1 and CPU2.

Instruction Type	A	B	C
Relative Frequency	35%	25%	40%
CPU1 CPI_i	1	1	3
CPU2 CPI_i	4	2	2

Given that CPU2 is 3 times faster than CPU1 when running ProgramX, what is the relation between clock rate of CPU1 (i.e. R1) and clock rate of CPU2 (i.e. R2)?

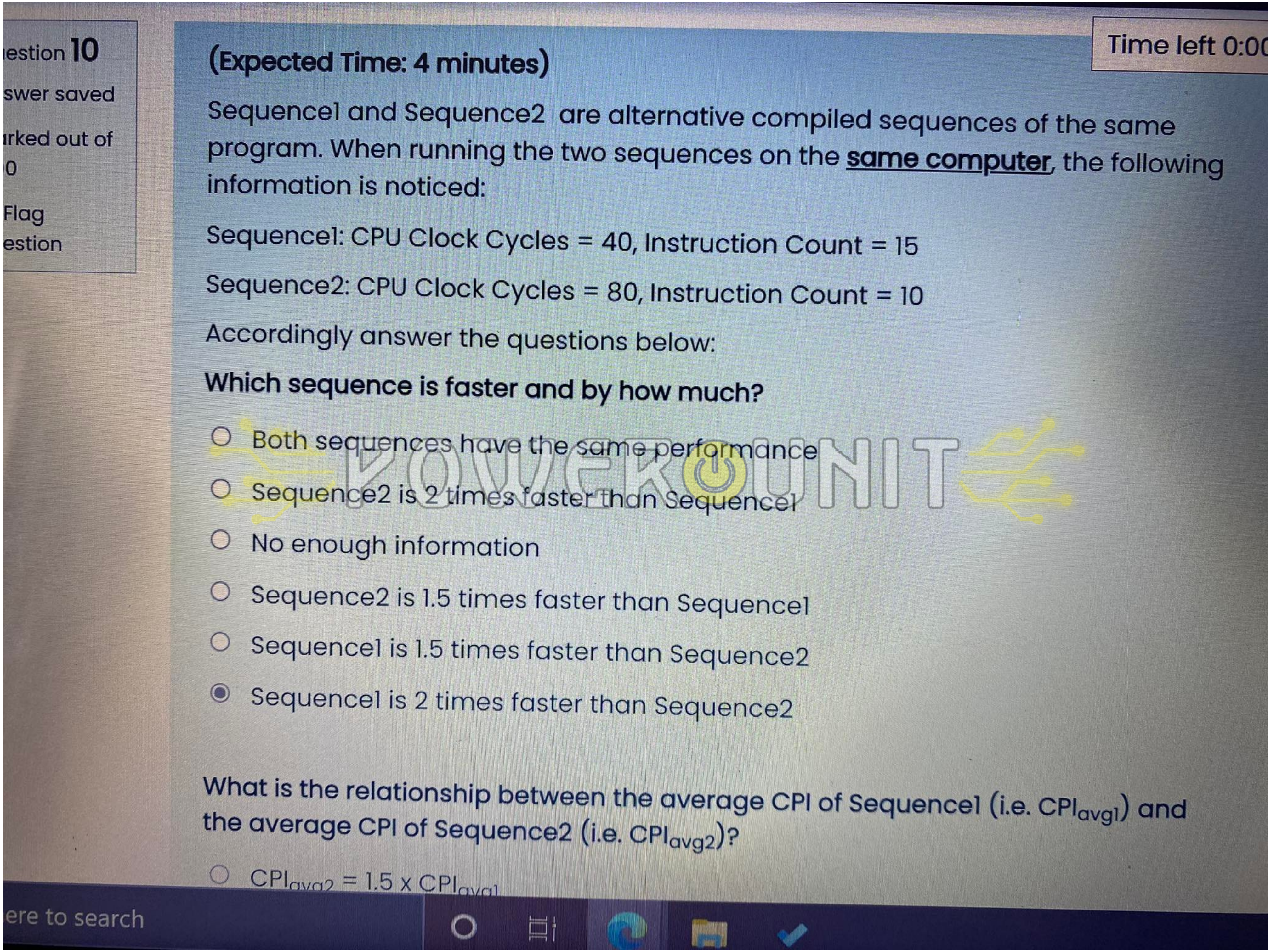
● $R2 = 4.5 \times R1$

○ $R2 = 4.8 \times R1$

○ $R1 = 4.8 \times R2$

○ $R1 = 3 \times R2$

○ $R2 = 3 \times R1$



(Expected Time: 4 minutes)

Sequence1 and Sequence2 are alternative compiled sequences of the same program. When running the two sequences on the **same computer**, the following information is noticed:

Sequence1: CPU Clock Cycles = 40, Instruction Count = 15

Sequence2: CPU Clock Cycles = 80, Instruction Count = 10

Accordingly answer the questions below:

Which sequence is faster and by how much?

○ Both sequences have the same performance

○ Sequence2 is 2 times faster than Sequence1

○ No enough information

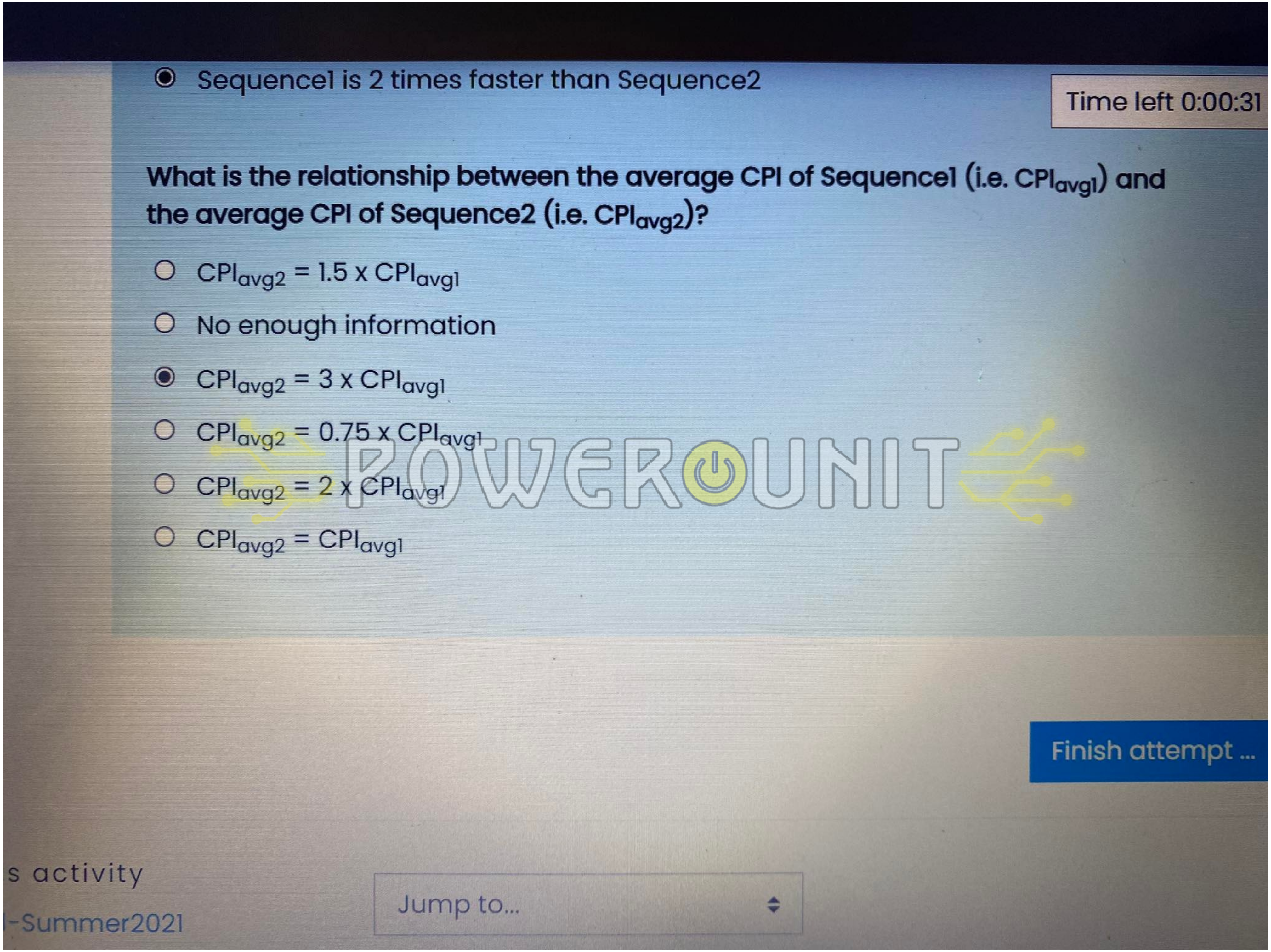
○ Sequence2 is 1.5 times faster than Sequence1

○ Sequence1 is 1.5 times faster than Sequence2

● Sequence1 is 2 times faster than Sequence2

What is the relationship between the average CPI of Sequence1 (i.e. CPI_{avg1}) and the average CPI of Sequence2 (i.e. CPI_{avg2})?

○ $CPI_{avg2} = 1.5 \times CPI_{avg1}$



● Sequence1 is 2 times faster than Sequence2

What is the relationship between the average CPI of Sequence1 (i.e. CPI_{avg1}) and the average CPI of Sequence2 (i.e. CPI_{avg2})?

○ $CPI_{avg2} = 1.5 \times CPI_{avg1}$

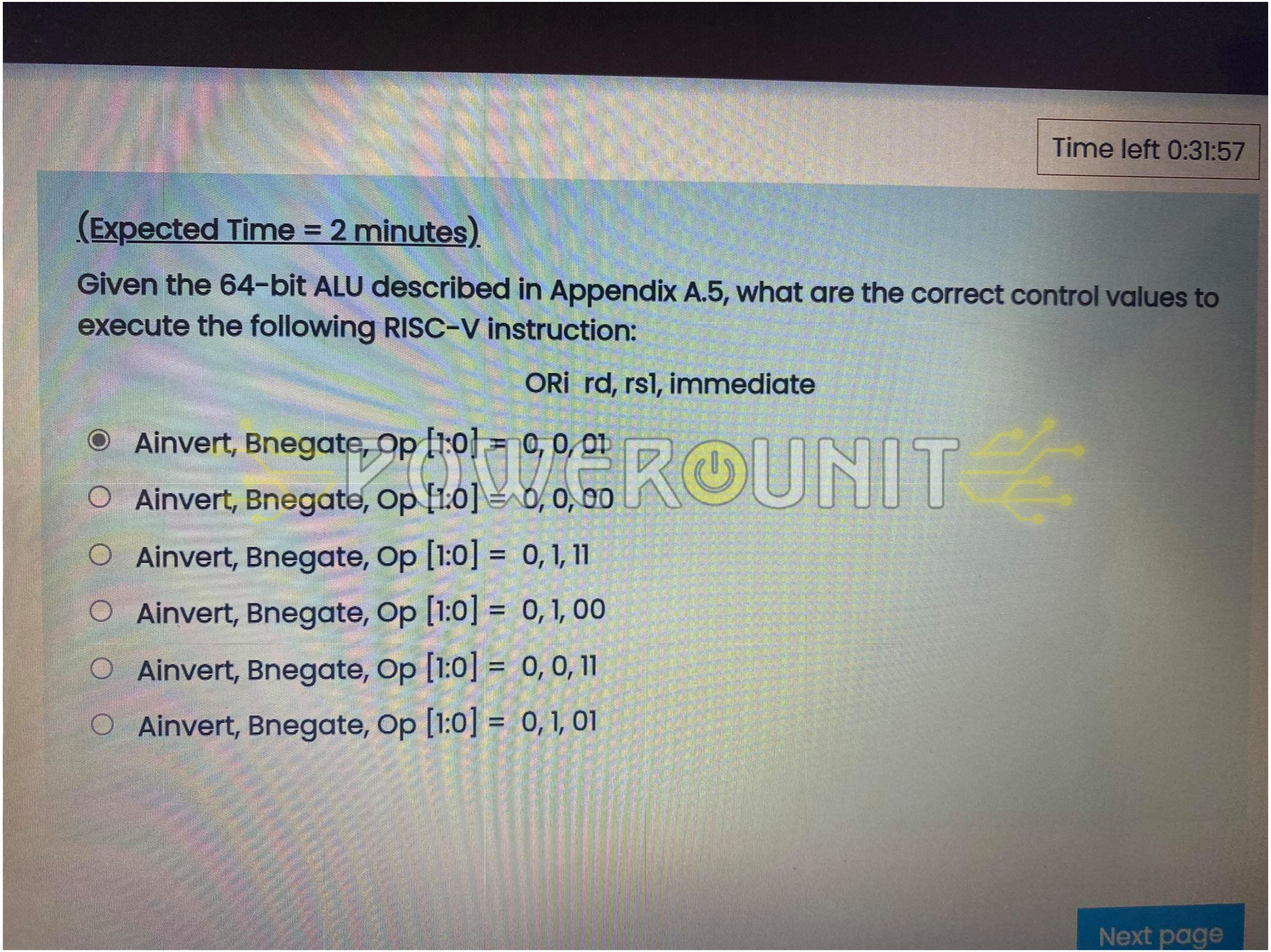
○ No enough information

● $CPI_{avg2} = 3 \times CPI_{avg1}$

○ $CPI_{avg2} = 0.75 \times CPI_{avg1}$

○ $CPI_{avg2} = 2 \times CPI_{avg1}$

○ $CPI_{avg2} = CPI_{avg1}$



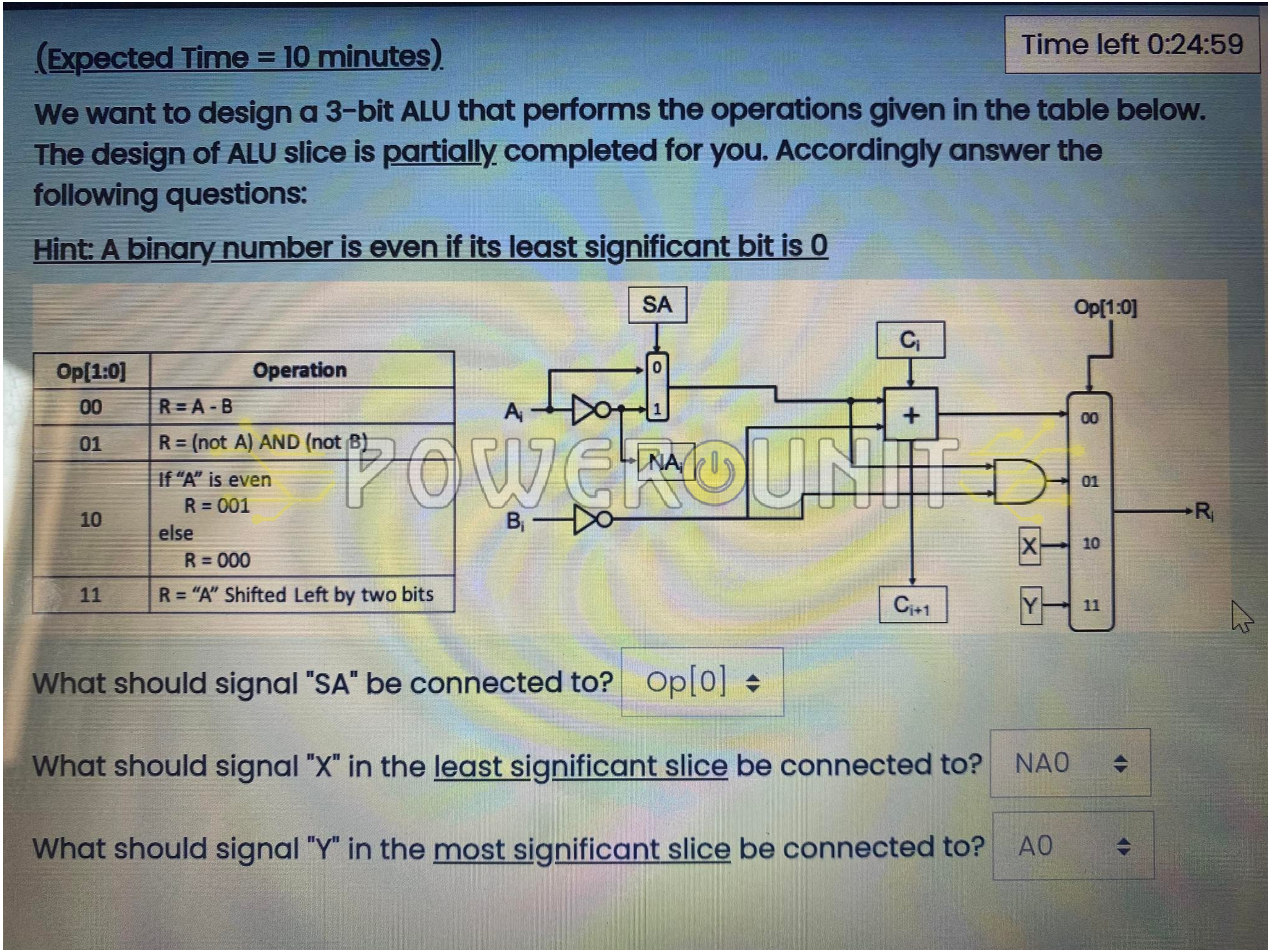
Time left 0:31:57

(Expected Time = 2 minutes)

Given the 64-bit ALU described in Appendix A.5, what are the correct control values to execute the following RISC-V instruction:

ORi rd, rs1, immediate

- ● Ainvert, Bnegate, Op [1:0] = 0, 0, 01
- ○ Ainvert, Bnegate, Op [1:0] = 0, 0, 00
- ○ Ainvert, Bnegate, Op [1:0] = 0, 1, 11
- ○ Ainvert, Bnegate, Op [1:0] = 0, 1, 00
- ○ Ainvert, Bnegate, Op [1:0] = 0, 0, 11
- ○ Ainvert, Bnegate, Op [1:0] = 0, 1, 01



Time left 0:24:59

(Expected Time = 10 minutes)

We want to design a 3-bit ALU that performs the operations given in the table below. The design of ALU slice is partially completed for you. Accordingly answer the following questions:

Hint: A binary number is even if its least significant bit is 0

Op[1:0]	Operation
00	R = A - B
01	R = (not A) AND (not B)

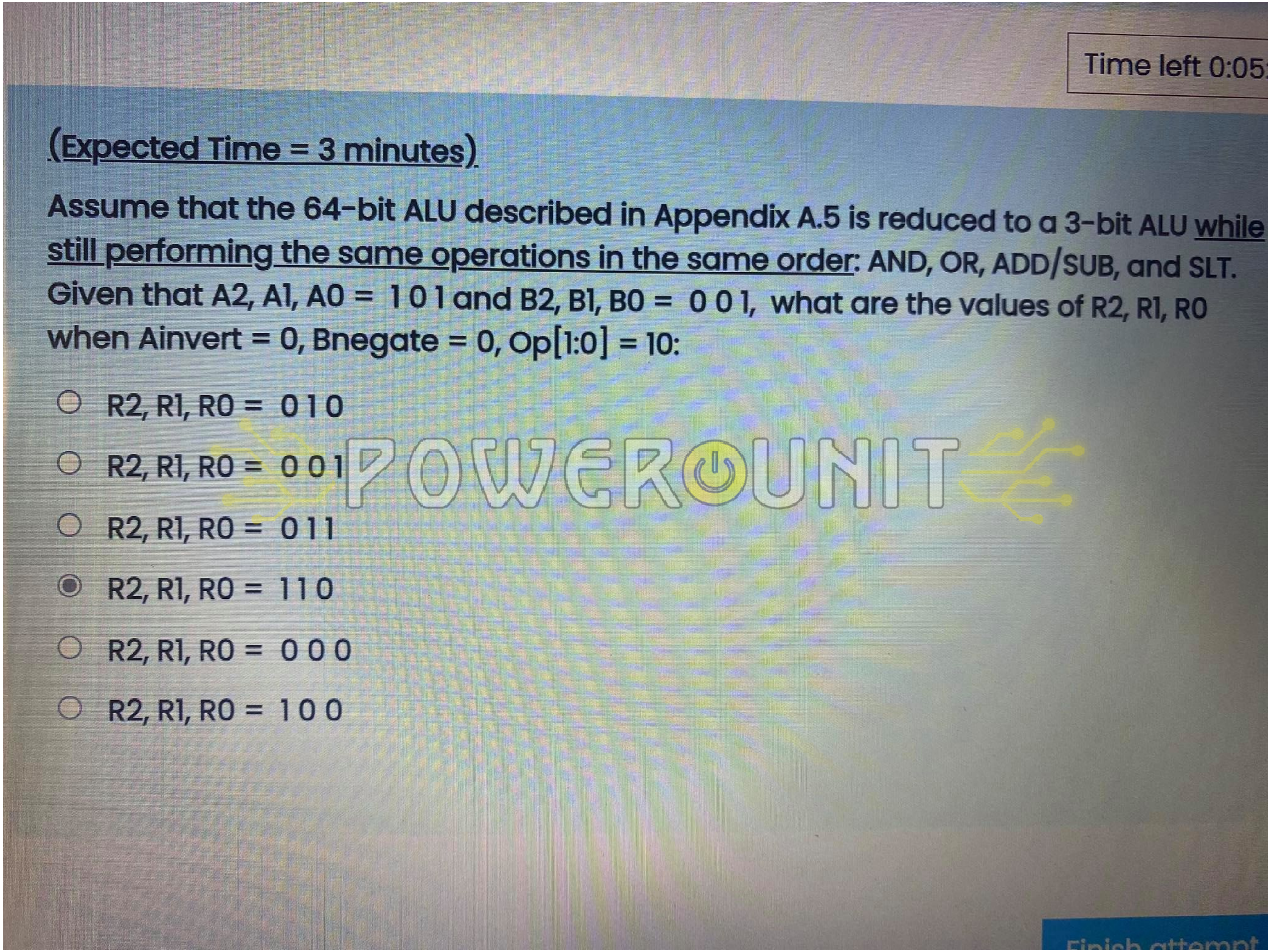
10	If "A" is even
	R = 001
	else
	R = 000

11	R = "A" Shifted Left by two bits

What should signal "SA" be connected to? Op[0] ⬍

What should signal "X" in the least significant slice be connected to? NA0 ⬍

What should signal "Y" in the most significant slice be connected to? A0 ⬍



(Expected Time = 3 minutes)

Assume that the 64-bit ALU described in Appendix A.5 is reduced to a 3-bit ALU while still performing the same operations in the same order: AND, OR, ADD/SUB, and SLT. Given that A2, A1, A0 = 1 0 1 and B2, B1, B0 = 0 0 1, what are the values of R2, R1, R0 when Ainvert = 0, Bnegate = 0, Op[1:0] = 10:

○ R2, R1, R0 = 0 1 0

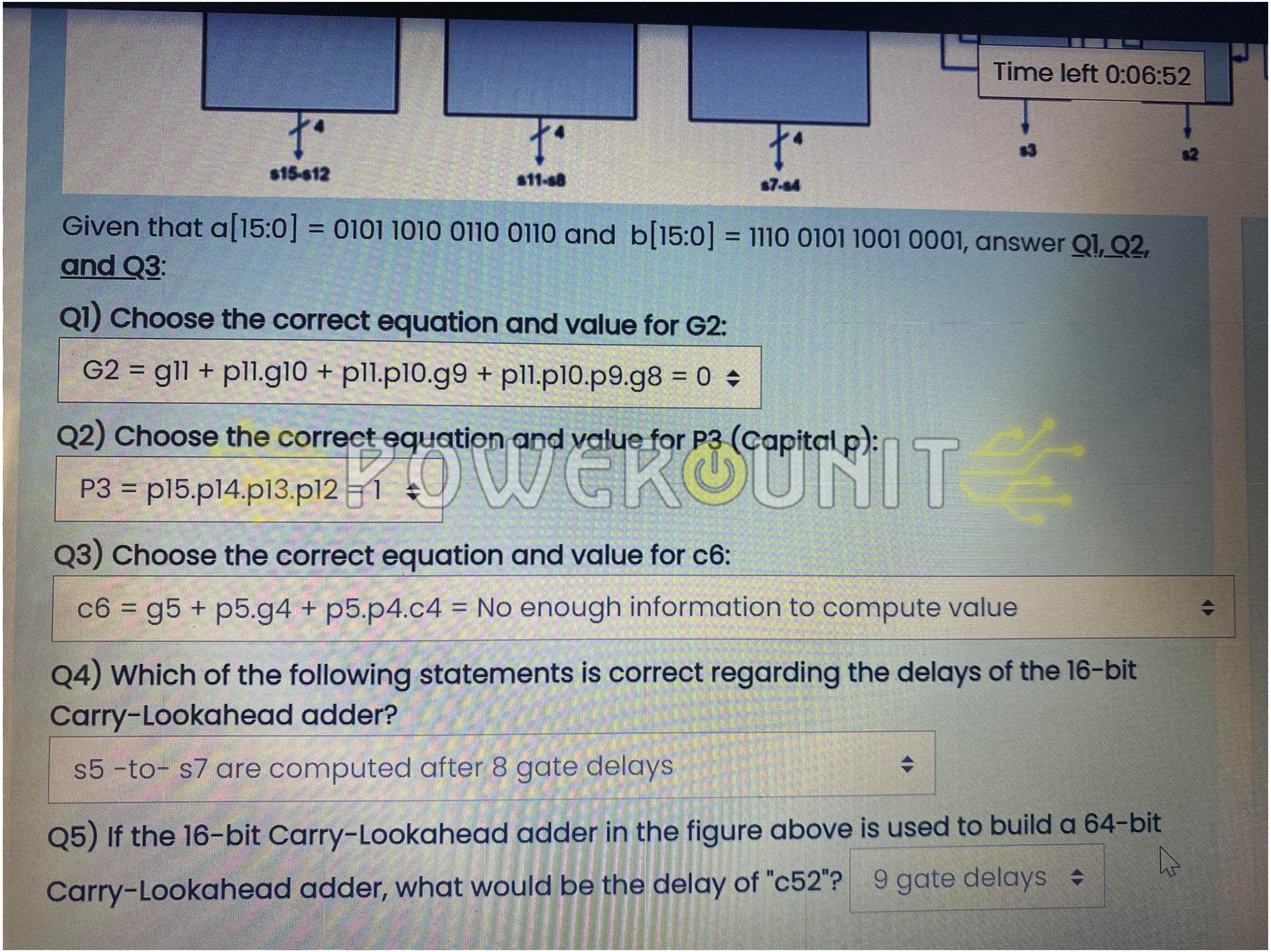
○ R2, R1, R0 = 0 0 1

○ R2, R1, R0 = 0 1 1

● R2, R1, R0 = 1 1 0

○ R2, R1, R0 = 0 0 0

○ R2, R1, R0 = 1 0 0



Given that a[15:0] = 0101 1010 0110 0110 and b[15:0] = 1110 0101 1001 0001, answer Q1, Q2, and Q3:

Q1) Choose the correct equation and value for G2:

G2 = g11 + p11.g10 + p11.p10.g9 + p11.p10.p9.g8 = 0 ⬍

Q2) Choose the correct equation and value for P3 (Capital p):

P3 = p15.p14.p13.p12 = 1 ⬍

Q3) Choose the correct equation and value for c6:

c6 = g5 + p5.g4 + p5.p4.c4 = No enough information to compute value ⬍

Q4) Which of the following statements is correct regarding the delays of the 16-bit Carry-Lookahead adder?

s5 -to- s7 are computed after 8 gate delays ⬍

Q5) If the 16-bit Carry-Lookahead adder in the figure above is used to build a 64-bit Carry-Lookahead adder, what would be the delay of "c52"? 9 gate delays ⬍