15/16

Questions 1 - 6: Each question has only one correct answer. Write your answers in the boxes provided in the Answer Booklet. One mark is awarded for a correct answer and no penalty for wrong answer.

1. When you need multi-bit values in Logisim, you use this tool as shown in the picture below. What is this tool called?

A. Splitter
B. Zipper
C. Comb
D. Brush
E. There isn't such a thing.

A

2. Which of the following is a **pseudo instruction** in MIPS?

A. **sub** (subtract)
B. **slti** (set less than immediate)
C. **bgt** (branch on greater than)
D. **xor** (bitwise exclusive-OR)
E. None of the above.

C

3. A program containing 5000 instructions is run on a machine with a clock frequency of 2 GHz. The table below shows the number of cycles for each instruction class and their frequencies in the program.

Instruction class	A	B	C	D
CPI	3	5	4	6
Frequency	40%	20%	20%	20%

How long does the program take to run on this machine?

A. 2.38 microseconds
B. 10.5 microseconds
C. 11.25 microseconds
D. 21.0 microseconds
E. 42.0 microseconds

$2000(3) + 1000(5) + 1000(4) + 1000(6)$
$= 21000$ cycles

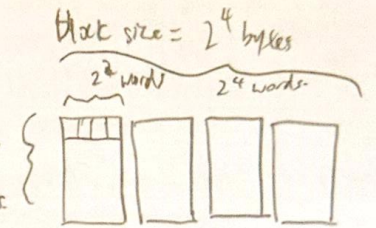
B

cache block size = 2^4 bytes cache set size = 2^6 bytes
No. of cache sets = 2^7 sets. offset = 4 bits, index = 7 bits.

2^{13} bytes = 2^{11} words

4. Consider a **4-way set associative cache** with a full data capacity of 8192 bytes. Each cache block consists of 4 words, and each word is 4 bytes long. What are the number of bits in the set-index field and the number of bits in the offset field of the memory address?

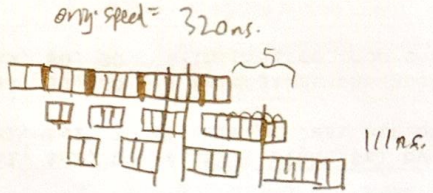
A. Set-index = 4 bits; Offset = 2 bits
B. Set-index = 6 bits; Offset = 2 bits
C. Set-index = 6 bits; Offset = 4 bits
D. Set-index = 7 bits; Offset = 4 bits
E. Set-index = 8 bits; Offset = 4 bits



D ✓

5. The five stages of a certain pipeline take 2 ns, 3 ns, 4 ns, 5 ns, and 2 ns. If there are 20 instructions, what is the maximum speedup in the execution time of a pipeline implementation compared to a single-cycle implementation?

A. 2.50
B. 2.67
C. 2.88
D. 3.20
E. 5.00

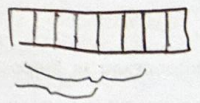


C

6. A certain machine has 3 types of instructions: *A*, *B* and *C*. Type-*A* instructions have opcode of 4 bits, type-*B* 6 bits, and type-*C* 8 bits. Assuming that each type must have at least one instruction, and the encoding space for opcode is completely utilized, what is the maximum number of type-*C* instructions you can have using an expanding opcode scheme?

A. 220
B. 227
C. 236
D. ~~254~~
E. ~~256~~

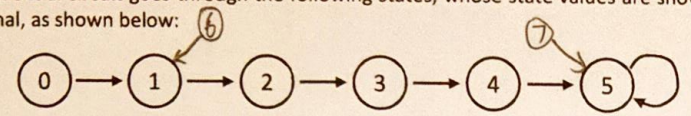
$2^8 - (2^6 - 1) - (2^4 - 1)$



$2^8 - 2^2 - 2^4$

C

7. **[8 marks]**
A sequential circuit goes through the following states, whose state values are shown in decimal, as shown below:



The states are represented by 3-bit values *ABC*. Implement the sequential circuit using a *JK* flip-flop for *A*, a *T* flip-flop for *B*, and a *D* flip-flop for *C*.

a. Write out the **simplified SOP expressions** for all the flip-flop inputs. Note that the simplified expression for *KA* has been done for you (*KA* = 0). [3 marks]

b. Complete the logic diagram on the answer booklet, by adding <u>one inverter</u> and a <u>minimum number of logic gates of another type</u>. [2 marks]

c. Complete the given state diagram on the answer booklet, by indicating the next state for each of the two unused states. [2 marks]

d. Is the circuit self-correcting? Explain your answer. (No mark will be awarded if there is no explanation or the explanation is wrong.) [1 mark]

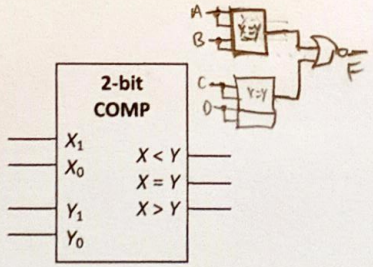
8. **[5 marks]**
Given the following Boolean function:

$$F(A,B,C,D) = \Sigma m(5, 6, 9, 10)$$

You are to implement *F* using <u>at most two 2-bit **magnitude comparators**</u> and <u>one two-input logic gate</u>. Note that complemented literals are not available.

No marks will be given if the above conditions are not met.

The block diagram of a 2-bit magnitude comparator is shown on the right. $X=X_1X_0$ and $Y=Y_1Y_0$ are unsigned binary values.



9. **[6 marks]** Answer the following parts about the **addi** (add immediate) instruction.

a. What are the values of the control signals **RegDst** and **ALUSrc** for **addi**? [2 marks]

b. Given that **$s1** contains the value 4, **$t1** contains the value 8, and the data in some memory are shown in the table on the right.

Address	*Data*
0	57
4	-43
8	100
12	3
16	98
20	62
24	-31

The **addi** instruction below is to be executed. Suppose due to some hardware fault, the value 1 is erroneously generated for the control signal **MemtoReg**. What is the final value in **$s1** after the **addi** instruction is executed? Explain clearly. [4 marks]

```
addi $s1, $t1, 12
```


10. **[10 marks]**
Study the partial MIPS program below.

The input **$a1** contains 30 bits of data padded with two zeroes at the end (right-most two bits). The 30 bits are data collected in a half-hour period. Each data bit indicates whether the light is *off* (0) or *on* (1) during a period of one minute. The state of the light (*off* or *on*) may only change at the beginning of a one-minute period. For instance, if **$a1** contains the following data (only the first 8 bits are shown) 01110010... it means that the light is *off* in the first minute, *on* in the next three minutes, *off* in the next two minutes, *on* in the next minute, and so on.

The partial MIPS program below computes the number of times the light changed from *off* to *on* in that 30-minute period, that is, the number of times "01" appears in **$a1**.

(a) Write the instruction encoding in hexadecimal for the **add $a1, $v0, $0** and **srl $a1, $a1, 2** instructions. [2 marks]

(b) Complete the program on the Answer Booklet using <u>not more than 10 MIPS instructions</u>. You are NOT to change or add any instruction before the "Loop" label. [8 marks]

```
# register $a1 contains a 32-bit value to be read
# register $a2 is the answer: the number of "01" in $a1

main: li   $v0, 5          # code 5: read_int call
      syscall              # syscall to read int
      add  $a1, $v0, $0    # transfer int read into $a1

      srl  $a1, $a1, 2     # shift right $a1 by 2 bits
      andi $t1, $a1, 1     # extract last bit of $a1 to $t1

      add  $a2, $0, $0     # initialise the answer to 0
      addi $t9, $0, 30     # initialise loop counter to 30

Loop:
```

11. **[7 marks]**

In the MIPS code below, register **$a0** stores the starting address of an integer array *A*. Assume a 5-stage MIPS pipeline processor.

```
        addi $t1, $a0, 12    # I1
        lw   $t0, 12($a0)    # I2
Loop:   lw   $t2, -4($t1)    # I3
        sw   $t2, 0($t1)     # I4
        addi $t1, $t1, -4    # I5
        bne  $t1, $a0, Loop  # I6
        sw   $t0, 0($t1)     # I7
```

(a) What does the code do? [2 marks]

(b) If the first instruction executed is instruction **I1**, which is the seventh instruction executed? [1 mark]

(c) Fill in the timing chart assuming no data forwarding, and branch resolution is at stage 4. You need to fill in only the first 7 instructions executed. Also, calculate the total number of cycles taken by the code after all iterations. [2 marks]

(d) Fill in the timing chart assuming data forwarding, branch resolution is shifted to stage 2 and branch prediction is used with the assumption that the branch is to be taken. You need to fill in only the first 7 instructions executed. Also, calculate the total number of cycles taken by the code after all iterations. [2 marks]

12. **[8 marks]**

Study the code below. The code accesses 3 integer arrays *A*, *B*, and *C*, each with 32 elements. Each element takes up 32 bits, which is one word in the MIPS architecture.

```
int sum = 0;
for (int i=0; i<32; i++) {
   sum = sum + (A[i] * B[i]) - C[i];
}
```

The starting addresses of the arrays are shown below:

- Array *A*: starting address at 0x00000080
- Array *B*: starting address at 0xFFFF0040
- Array *C*: starting address at 0x12345688

Given a **direct-mapped cache** with 8 blocks, each block containing 4 words.

(a) What is the cache hit rate of the above code? You may write your answer as a fraction. [2 marks]

(b) Fill in the content of the cache after executing the above code. [6 marks]

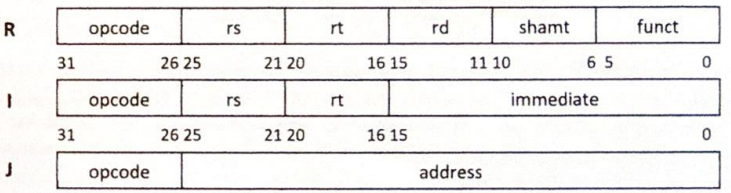
For instance, if you think that block 0 of the cache contains *A*[12], *A*[13], *B*[24] and *B*[25], you may fill in the cache this way:

Block 0	*A*[12]	*A*[13]	*B*[24]	*B*[25]

MIPS Reference Data (partial)

Name	Mnemonic	Format	Operation	Opcode/Funct
Add	add	R	R[rd] = R[rs] + R[rt]	$0/20_{hex}$
Add Immediate	addi	I	R[rt] = R[rs] + SignExtImm	8_{hex}
And	and	R	R[rd] = R[rs] & R[rt]	$0/24_{hex}$
Branch on Equal	beq	I	If (R[rs] == R[rt]) PC = PC + 4 + BranchAddr	4_{hex}
Load Word	lw	I	R[rt] = M[R[rs] + SignExtImm]	23_{hex}
Or	or	R	R[rd] = R[rs] | R[rt]	$0/25_{hex}$
Set Less Than	slt	R	R[rd] = (R[rs] < R[rt]) ? 1 : 0	$0/2A_{hex}$
Shift Right Logical	srl	R	R[rd] = R[rt] >> shamt	$0/02_{hex}$
Store Word	sw	I	M[R[rs] + SignExtImm] = R[rt]	$2B_{hex}$
Subtract	sub	R	R[rd] = R[rs] - R[rt]	$0/22_{hex}$

BASIC INSTRUCTION FORMATS



REGISTER NAMES AND NUMBERS

Name	Number	Use
$zero	0	The constant value 0
$at	1	Assembler Temporary
$v0 – $v1	2 – 3	Values for Function Results and Expression Evaluation
$a0 – $a3	4 – 7	Arguments
$t0 – $t7	8 – 15	Temporaries
$s0 – $s7	16 – 23	Saved Temporaries
$t8 – $t9	24 – 25	Temporaries
$k0 – $k1	26 – 27	Reserved for OS Kernel
$gp	28	Global Pointer
$sp	29	Stack Pointer
$fp	30	Frame Pointer
$ra	31	Return Address

~~ END OF PAPER ~~~