*Wednesday 22-08-2012 15:00-16:30
HPH G1, HPH G2, HPH G3

Name :
First name :
Student ID:

1st session examination

Answers in Red

Design of Digital Circuits SS12

(252-0014-00S)

S. Capkun, F.K. Gurkaynak

Examination rules:

1. written 90 minutes
2. No books, no calculators, no computers or communication devices. 5 pages handwritten notes are allowed.
3. Write your answers on this document, space is reserved for your answers after the questions, if you write your answers somewhere else (e.g. draft pages) specify it.
4. Put your student card visible on the desk during the exam.
5. Immediately call an assistant, if you feel disturbed.
6. Answers will only be evaluated if they are readable.
7. Write with black or blue ink (no pencil, no green or red color).
8. Show your work. For some questions, you may get partial credit even if the end result is wrong due to a calculation mistake

Question	Total Points	Points
1	5	
2	5	
3	15	
4	10	
5	10	
6	10	
7	10	
8	10	
Total	75	

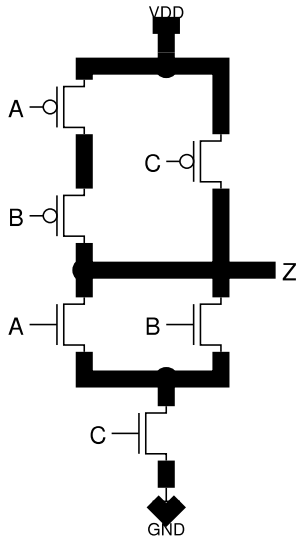
1.

a) Below you can see on the left four binary numbers and on the right 4 interpretations of these numbers and a corresponding value. Match the number on the left to the descriptions on the right. (2 points)

Binary Number		Value, Interpretation	
1	11110	A	Decimal -1, 5-bit two's complement
2	10001	B	Decimal -1, 5 bit sign magnitude
3	10010	C	Decimal 30, 5 bit unsigned
4	11111	D	Hexadecimal 0x12, 5 bit unsigned

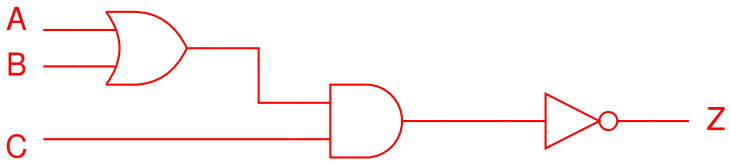
1-C 2-B 3-D 4-A

b) Consider the transistor level schematic below. What is the output going to be when A=1, B=0, C=1? (1 point)



0

c) Using only 2-input AND, 2-input OR, or inverters, draw a gate level schematic that realizes the same Boolean Function as the circuit shown in 1(b). (2 points)



2.

a) Write the truth table for the function Z= B'(C' + A) + BC'. (1 point)

A	B	C	Z
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

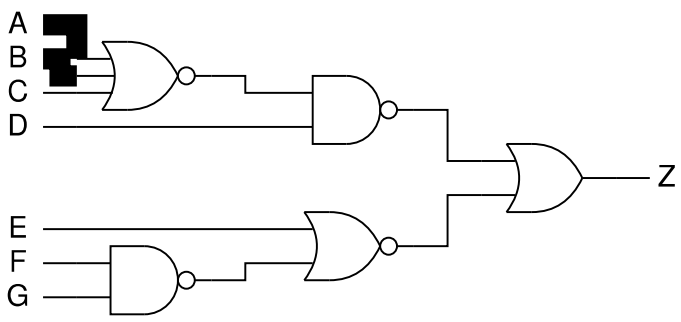
b) Compose a Karnaugh Map for the truth table from question 2(a). (1 point)

A\BC	00	01	11	10
0	1	0	0	1
1	1	1	0	1

c) Find a minimal Boolean Equation from the Karnaugh Map (2(b)) or the Boolean Equation (2 (a)) for Z (1 point)

Z = AB' + C'

d) Examine the circuit below. We want to find out the Boolean equation by inspection. You can use bubble-pushing methods to simplify the circuit. Write the Boolean equation. (2 points)

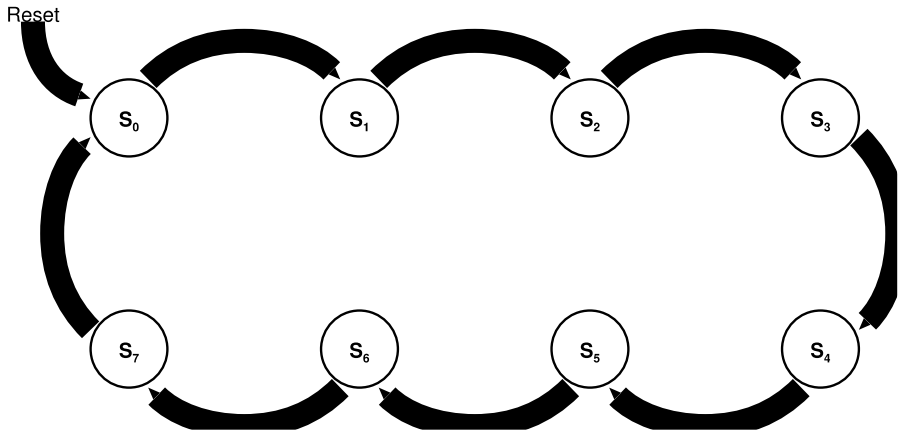


Z = A + B + C + D' + E'FG

3. In this question you will design a simple Finite State Machine (FSM) that implements a 3-bit Gray Code counter. The FSM will not have any inputs and have three output bits G_2, G_1, G_0. Gray codes are specialized codes where consecutive numbers differ in only one bit position as seen in the table below.

State	**Gray Code Output**		
S	G_2	G_1	G_0
S_0	0	0	0
S_1	0	0	1
S_2	0	1	1
S_3	0	1	0
S_4	1	1	0
S_5	1	1	1
S_6	1	0	1
S_7	1	0	0

The following is a state transition diagram of this FSM with the states named S_0 to S_7.



a) Is this a Moore or Mealy type FSM? (1 point)

Moore type, since the output G only depends on the state as there are no inputs

b) It has been decided to use a simple binary state encoding using 3-bits where each state is encoded in standard binary. I.e. $S_0 = 000$, $S_1 = 001$, … , $S_6 = 110$, $S_7 = 111$. Make a state transition table using the binary state encodings given above. Determine the next state equations. (4 points)

Current State			**Next State**		
S_2	S_1	S_0	N_2	N_1	N_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

N2 = S2'S1S0 + S2S1' + S2S0'
N1 = S1'S0 + S1S0'
N0 = S0'

c) Now determine the output equations that calculate the outputs G2, G1, G0 from the state bits S0, S1, S2. (3 points)

G2 = S2
G1 = S2S1' + S2'S1
G0 = S1'S0 + S1S0'

d) In step b), we have used a binary coding for the states. As a result, we needed a additional circuit to calculate the outputs from the State bits. In this part we will directly use the gray code for the state encoding. In this way, the state will directly be the required gray code, and no additional output encoding will be required.

Make a new state transition table and determine the next state equations using the gray code as state encoding. (5 points)

Current State			**Next State**		
S_2	S_1	S_0	N_2	N_1	N_0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	0	1

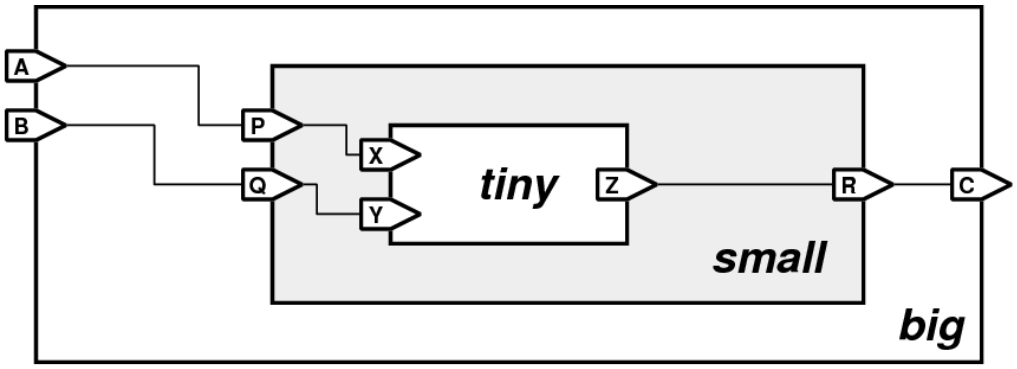
N2 = S2S0 + S1S0’
N1 = S2’S0 + S1S0’
N2 = S2’S1’ + S2S1

e) Which solution would you prefer (using binary coding for the states as in b or gray coding as in d)? Explain with a short sentence. (2 points)

Using the gray code as the state encoding results in a simpler circuit with fewer gates, it would be better to use that one.

4. In this question for each part there will be two Verilog code snippets. For each part you will have to say whether both, only one, or none of the code snippets fulfill what is being asked. All code snippets are syntactically correct. They will compile and produce either a sequential circuit or a combinational circuit. (2 points each)

a) Which code snippet(s) realizes the following hierarchy of three instances given in the figure below? (Note the function "tiny" realizes a simple AND function)



Code Snippet (A)	Code Snippet (B)

`module big (input A,B,`
`output C);`
`module small (input P,Q`
`output R);`
`module tiny (input X,Y,`
`output Z);`
 	
`assign Z = X & Y;`
`assign R = Z;`
`assign C = R;`
 	
`endmodule`
`endmodule`
`endmodule`
	`module tiny (input X,Y,`
	`output Z);`
	`assign Z = X & Y;`
	`endmodule;`
	
	`module small (input P,Q`
	`output R);`
	`tiny tim (P,Q,R);`
	`endmodule`
	
	`module big (input A,B,`
	`output C);`
	`small sam (A,B,C);`
	`endmodule`

☐ Only A ☐ Only B ☐ Both A and B ☐ None

b) Which code snippet(s) will produce a four input multiplexer?

Code Snippet (A)	Code Snippet (B)

`assign z = sel[0] ?`
`(sel[1] ? c : d )`
`:`
`(sel[1] ? b : a);`
	`always @ (*)`
	`case (sel)`
	`2'b00: z = a;`
	`2'b10: z = b;`
	`2'b11: z = c;`
	`default: z = d;`
	`endcase`

☐ Only A ☐ Only B ☐ Both A and B ☐ None

c) Which code snippet(s) will produce a 8-bit value which is composed of (from MSB to LSB), $c_2c_1d_0d_0d_0001$ (c and d are both 8-bit values)?

Code Snippet (A)	Code Snippet (B)

`always @ (*)`
`  begin`
`  z  <= 8'b00000001;`
`  c2 <= c << 6;`
`  d2 <= d << 3;`
`  z  <= z & c2 & d2;`
`  end`
	`assign z = { c[2:1],`
	`  {3{d[0]}},`
	`  3b'001};`

☐ Only A ☐ Only B ☐ Both A and B ☐ None

d) Which code snippet(s) will produce a sequential circuit?

Code Snippet (A)	Code Snippet (B)

`always @ (some, signal)`
`  if (signal)`
`  lone <= some;`
	`always @ (posedge clk)`
	`  en <= data;`

☐ Only A ☐ Only B ☐ Both A and B ☐ None

e) Which code snippet(s) will produce a falling edge triggered D-type flip-flop with an asynchronous reset?

Code Snippet (A)	Code Snippet (B)

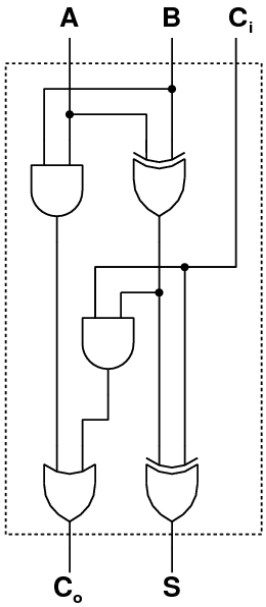
`always @ (posedge clk)`
`  if (reset) q <= 1'b0`
`  else  q <= data;`
	`always @ (negedge clk)`
	`  if (reset) q <= data;`

☐ Only A ☐ Only B ☐ Both A and B ☐ None

5. In this question we will compute the Area and Delay of different adder components. To calculate the Area and the Speed use the values in the following table:

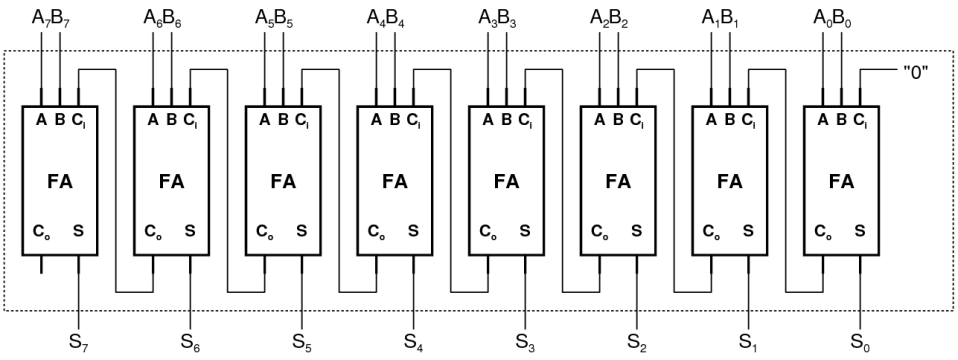
Gate	Delay (all paths)	Area
2-input AND	15ps	1.8 μm^2
2-input OR	15ps	1.8 μm^2
2-input XOR	20ps	2.3 μm^2

a) The figure below is a gate level schematic of a 1-bit full adder. Using the table above: Determine the total area of the 1-bit full adder, identify the critical path in this circuit by drawing on the schematic, and calculate the critical path using the table. (3 points)



A_{FA} = 2.3µm^2+ 2.3µm^2 + 1.8µm^2 + 1.8µm^2 + 1.8µm^2 = 10µm^2
Critical path from A/B to Co
t_{crit}= t_{XOR} + t_{AND} +t_{OR} = 20ps + 15ps + 15ps = 50ps

b) An 8-bit Ripple Carry Adder is generated from the 1-bit Full Adder from the previous question 5a. If this adder is used to add 8-bit two's complement numbers, what is the total area and the critical path of this 8-bit adder? (3 points)

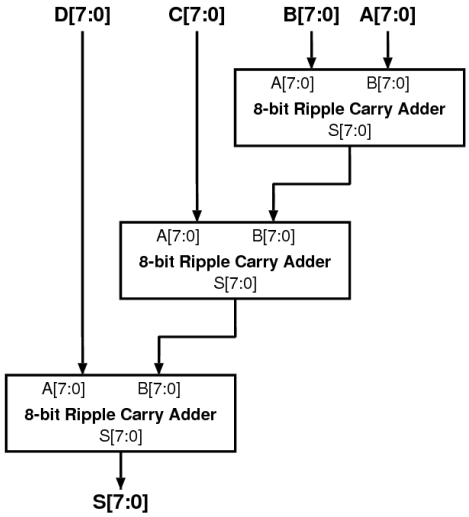


$A_{Tot} = 8 \times A_{FA} = 80\ \mu m^2$

T_{crit} is a little tricky. The C_i for the LSB is 0. So the signal there propagates through a shorter path (One AND and one OR gate =30ps),
Since only Two's complement numbers are used, the carry out S8 is not used, For the MSB, only A/B to S delay is relevant = 40ps
$T_{crit}=t_{MSB} + 6 \times t_{FA} + 1 \times t_{LSB} = 40+ 6 \times 50ps + 30ps = 370ps.$

$8 \times t_{FA} = 400ps$ is also acceptable should give them -1 point

c) A multi-operand adder to add four 8-bit two's complement numbers is constructed using the 8-bit ripple carry adder structure from the question 5b as shown in the figure below. What is the total area and the critical path of this multi-operand adder? (4 points)



Total Area = 3x Eightbit RCA = $3 \times 80\ \mu m^2 = 240\ \mu m^2$

The timing is trickier. The critical path goes through the LSB of the first adder, and then the second LSB to the S_0 outputs (40ps each). Then you have the normal critical path of the eight-bit RCA calculated in the previous question (370ps). Together it is $T_{crit} = t_{B,S} + t_{B,S} + T_{8bitRCA} = 40ps + 40ps + 370ps = 450ps.$

Note: 1ps = 0.000 000 000 001s = 1.10^{-12}s
$1\mu m^2$ = 0.000 000 000 001m^2 = 1.10^{-12} m^2

6. This exercise uses MIPS assembly instructions. The relevant entries from the Appendix B of your book are given for the instructions used in this exercise.

Given below is an assembly program to perform a certain operation. Go through the program step by step to answer the following questions.

A MIPS Assembly Program

```
begin:  addi $t1, $0, 0
        addi $t2, $0, 1

loop:   slt $t3, $t5, $t2
        bne $t3, $0, output
        add $t1, $t1, $t2
        addi $t2, $t2, 2
        j loop

output: add $t6, $t1, $0
```

a) What does the above MIPS assembly program do? What is the value stored in output register $t6 at the end of program execution if the input register $t5 contains the decimal value 10? (3 points)

b) Modify the program to load input from memory address 0x00000010 and store the output in memory address 0x00000020 instead of the registers $t5 and $t6. (2 points)

c) For reusability of code, we rewrite the assembly program given in (a) using subroutines (procedures). The functionality of the code remains the same. Complete the modified assembly code below by filling in the empty blocks. (2 points)

Assembly program using subroutines

```
begin     : add $a0, $t1, 10 # $t1 is the input reg
            jal function
            add $t6, $v0, $0
halt      : j  halt

function  : addi $t1, $0, 0
            addi $t2, $0, 1

   loop   : slt $t3, $a0, $t2
            bne $t3, $0, exit_func
            add $t1, $t1, $t2
            addi $t2, $t2, 2
            j loop

exit_func : add $v0, $t1, $0
            jr $ra
```

d) What is the value stored in register $t1 at the end of program execution for the code given in (c)? (1 point)

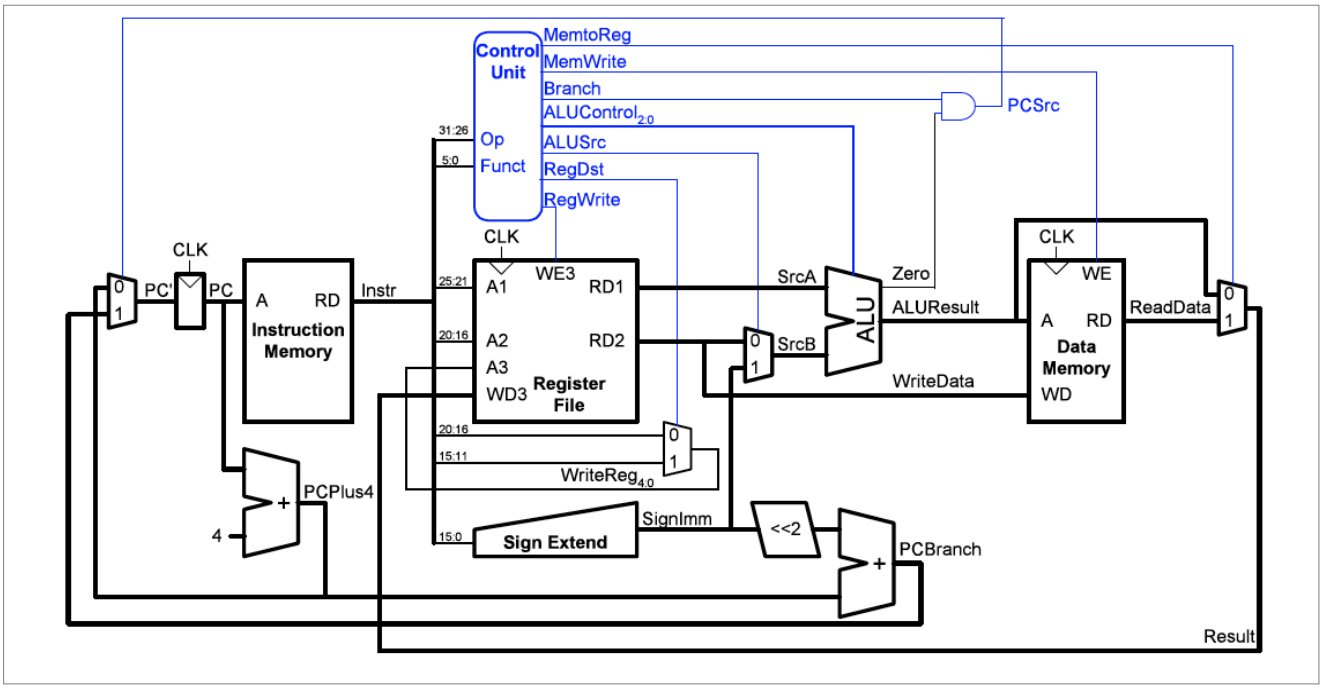
e) As you can observe that the subroutine *function* overwrites register $t1, suggest modifications to the code to preserve $t1's contents. (2 points)

Relevant entries from Appendix B

`[reg]:`	contents of register
`SignImm:`	sign-extended immediate `={{16{imm[15]}},imm}`
`[Address]:`	contents of memory location `Address`
`BTA:`	branch target address `= PC + 4 + SignImm <<2)`
`JTA:`	jump target address `= {(PC+4)[31:28], addr, 2'b0}`

Name	Description	Operation
`j`	Jump	`$ra = PC +4, PC =JTA`
`jal`	Jump and link	`$ra = PC +4, PC =JTA`
`beq`	Branch if equal	`If ([rs] == [rt]) PC = BTA`
`addi`	Add immediate	`[rt] = [rs] + SignImm`
`lw`	Load word	`[rt] = [Address]`
`sw`	Store word	`[Address] = [rt]`
`jr`	Jump register	`PC = [rs]`
`and`	And	`[rd] = [rs] & [rt]`
`xor`	Xor	`[rd] = [rs] ^ [rt]`

7. The following is a diagram of a single cycle MIPS architecture that is able to execute R-type and I-type instructions.



a) Determine the value of the control signals when this architecture executes a **beq** instruction, and fill in the table below. Note that the ALU can be programmed to perform the following functions: addition, subtraction, and, or. (3 points)

Control Signal	**Value**
RegDst	X
ALUSrc	0
MemWrite	0
MemtoReg	X
RegWrite	0
Branch	1
AluOperation (Add/Sub/And/Or)	Sub

b) Draw the data flow on the block diagram above (2 points)

c) Briefly explain the advantages of a multi-cycle architecture when compared to the single-cycle architecture shown above. (3 points)

In a single-cycle architecture, all instructions are given 1-cycle to execute, therefore the slowest instruction determines the speed of the processor.

In a multi-cycle processor, instructions are broken down into smaller pieces, decreasing the cycle time. Simpler instructions can be executed faster, reducing the average cycle time.

A single cycle processor, needs multiple instances of memories, and adders which may be quite large. A multi-cycle processor can share these resources, using only a single memory and ALU. This reduces the area

d) Which of the following statements about microarchitectures are **TRUE** (Mark all that apply)? (2 points)

In a pipelined architecture, a given instruction is executed faster than in a single-cycle architecture. (FALSE, a given instruction runs even slightly slower, due to the overhead, but the throughput increases)

In a pipelined architecture, control hazards can occur following the branch instruction, since the next instruction address may not be determined in time. (TRUE)

The Clocks per Instruction (CPI) of a micro-architecture is calculated as a weighted average of instructions executed in a given program/benchmark, and therefore is program dependent. (TRUE)

A multi-cycle architecture has less control overhead than a single-cycle architecture. (FALSE, first there are more resources to be shared, and there is overhead for the sequential processing)

8. In this exercise we will evaluate the memory access time of a small program under different cache configurations. The program will access the following 20 addresses in order (addresses are given as 8-bit hex numbers for simplicity):

```
0x00 0x04 0x08 0x0C 0x00 0x04 0x10 0x14 0x40 0x44→
0x00 0x04 0x48 0x4C 0x08 0x0C 0x00 0x04 0x48 0x4C
```

In this system one main memory access takes 20ns.

a) If the system has no cache, how much time will it make all memory accesses in the program given above? (1 point)

t_{total} = N x t_{mem}.
t_{total} = 20 x 20ns. = 400 ns

b) As an alternative, it was decided to use a direct mapped cache with capacity of 8 words and a block size of 1. The cache access time for this cache is 2ns. Using the table below, show the final content of this cache memory **after** executing the program above. (2 points)

Location	Content
Set 7	
Set 6	
Set 5	14
Set 4	10
Set 3	~~0C~~ ~~4C~~ ~~0C~~ 4C
Set 2	~~08~~ ~~48~~ ~~08~~ 48
Set 1	~~04~~ ~~44~~ 04
Set 0	~~00~~ ~~40~~ 00

c) How many compulsory cache misses were there? (1 point)

There are six compulsory misses: the first four accesses to 00 04 08 0C and then the accesses to 10 14 on the 7th and 8th cycles.

d) How many conflict misses were there? (1 point)

There are 10 conflict misses: 8th cycle 40 conflicts with 00, 9th cycle 44 conflicts with 04, 10th cycle 00 conflicts with 40, 11th cycle 04 conflicts with 44, 12th cycle 48 conflicts with 08, 13th cycle 4C conflicts with 0C, 14th cycle 08 conflicts with 48, 15th cycle 0C conflicts with 4C,18th cycle 48 conflicts with 08, 19th cycle 4C conflicts with 0C

e) What is the Miss Ratio for this cache? (1 points)

There are 16 misses out of 20 accesses. So the Miss Rate is 16/20 = 80%

f) How long will it take to make all the memory accesses for the program given above? (2 points)

There are 20 cache accesses each 2ns = 2 x 20ns = 40ns
There are 16 cache misses, each resulting in a memory access=16x20ns=320ns
Total is 40ns + 320ns = 360ns
OR = AMAT = t_{cache} + (MR x t_{mem}) = 2ns + (0.8 x 20ns) = 18ns.
Total time memory access x AMAT = 20 x 18ns = 360ns

g) There are four suggestions below. In each case only one parameter of the cache will be changed. Which of the following changes would improve the total memory access time **of this system running the above program**, indicate all that apply? (2 points)

a. Increasing the Capacity from 8 to 16
b. Increasing Block size from 1 to 2
c. Increasing Set Associativity from 1 (direct mapped) to 2
d. Increasing Cache Access Time 1ns to 2 ns

Note: 1ns = 0.000 000 001s = 1.10^{-9}s
1MHz = 1 000 000 Hz = 1.10^{6} Hz