

NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

EXAMINATION FOR

Semester 1 AY2010/11

CS2100 – COMPUTER ORGANISATION

Nov 2010

Time allowed: 2 hours

Your Matriculation Number:

--

INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of **SIX (6)** questions and comprises **TWENTY-ONE (21)** printed pages including this page.
2. This is an **OPEN BOOK** examination. You may use any approved calculators but not any PDA or laptop, especially those capable of external connectivity or communication.
3. Answer all questions. Note that the full mark for each question is different.
4. Write your answers on *this* **QUESTION AND ANSWER SCRIPT**. Answer only in the space given. Any writing outside this space will not be considered. No other submission is allowed.
5. Fill in your Matriculation Number with a pen, clearly on every page of this **QUESTION AND ANSWER SCRIPT**.
6. You may use pencil to write your answers.
7. At the end of the examination, please check to ensure that your script has all the pages properly stapled together.
8. Note that when a number is written as “0xNNNN” it means that “NNNN” is in base 16.

Total Score

--

/100

QUESTION 1 (15 marks)

- (1a) Encode the MIPS instruction “**lui \$5, 0xFEED**”, leaving your answer as an 8-hexadecimal digit number. (2 marks)

ANSWER:

- (1b) Encode the MIPS instruction “**beq \$t1, \$s0, Label**”, leaving your answer as an 8-hexadecimal digit number. Assume that the PC of this instruction is 0x1000 and Label is located at address 0x880. (2 marks)

ANSWER:

(1c) What is the MIPS instruction that is encoded by the hexadecimal number
0x00016C42? (2 marks)

ANSWER:

(1d) Convert following C code snippet into MIPS assembly.

```
sum = 0;
for (i=0; i<N; i++) {
    if (A[i] == i) continue;
    sum += A[i];
}
```

Note that: sum is in \$s0, address of the word (4-byte) array A is in \$a0, i is in \$t0, and N is in \$s1. (6 marks)

ANSWER:

(1e) Consider the following C code:

```
if (x > y) goto FARFARAWAY;
```

Suppose x is in $\$t0$, and y is in $\$t1$. The PC of this instruction is $0x100$ and FARFARAWAY is located at address $0xEEEE1800$. Give the MIPS instructions that will implement this if-statement. (3 marks)

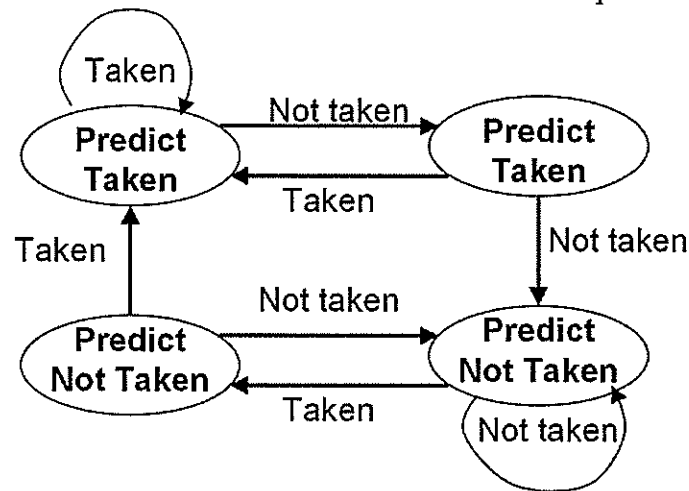
ANSWER:

QUESTION 2 (15 marks)

- (2a) The MIPS beq/bne requires the comparison of two 32-bit quantities. If it is performed at the ALU, subtraction would be used. However, when we move it to the ID stage, we need a specialized comparator unit that compares the two quantity and outputs a single bit result indicating whether the two quantities are equal or not. There is no need for subtraction. Using basic gates, construct a circuit that will perform such a comparison. Assume that any gate has a maximum fan-in of 4 inputs. (5 marks)

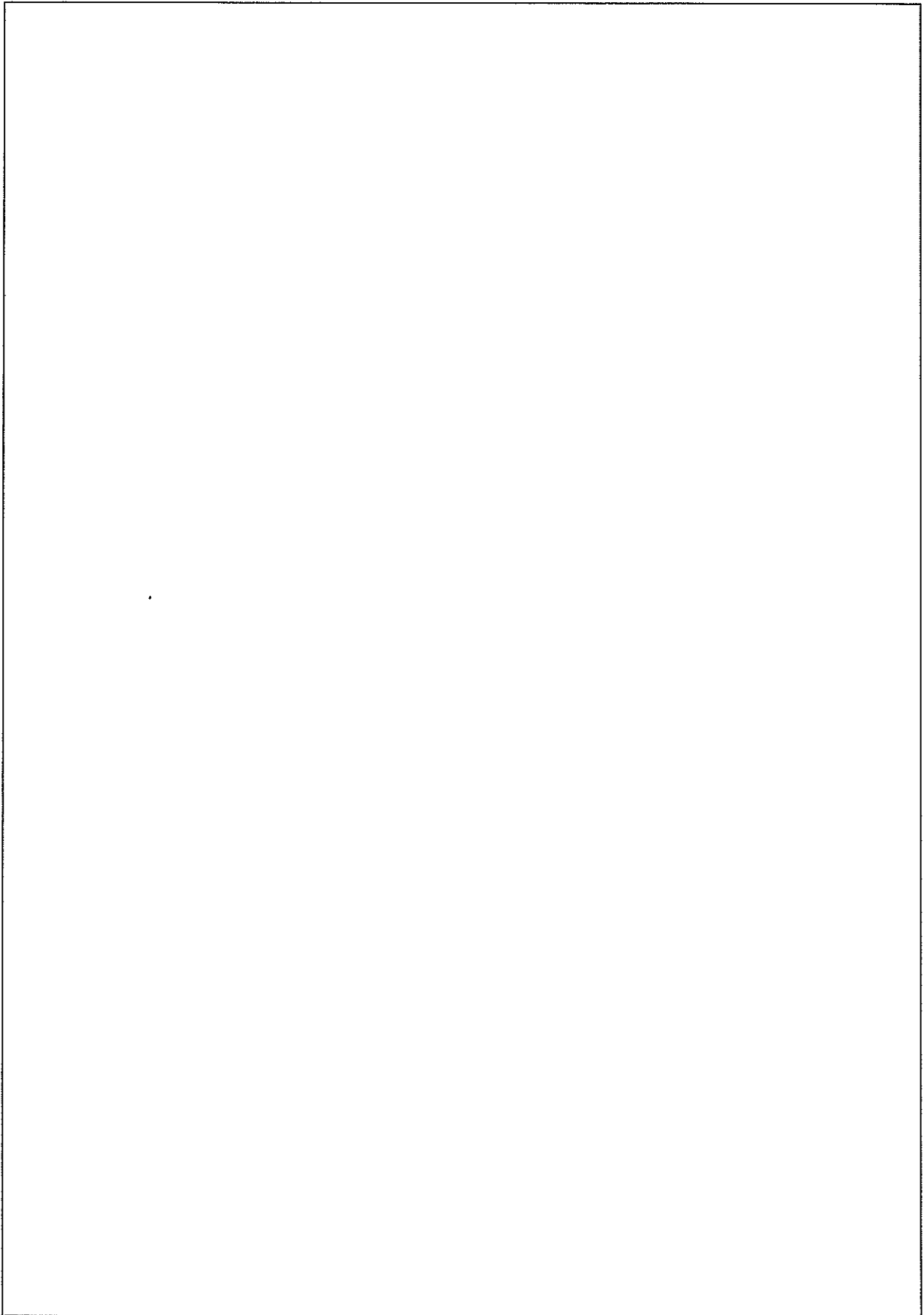
ANSWER:

(2b) The following is the finite state machine for the 2-bit branch predictor.



Implement this finite state machine (including the prediction output) using two D-flip-flops, and the standard gates. The “taken/not-taken” (“0” = “not taken”, “1” = “taken”) is a single bit of input. The prediction, i.e., “Predict Taken/Not Taken” is a 1-bit output. Show clearly, how you encode the 4 states and your intermediate workings, including the truth table from which you derived the implementation. You may assume that the flip-flops have a fixed clock input. (10 marks)

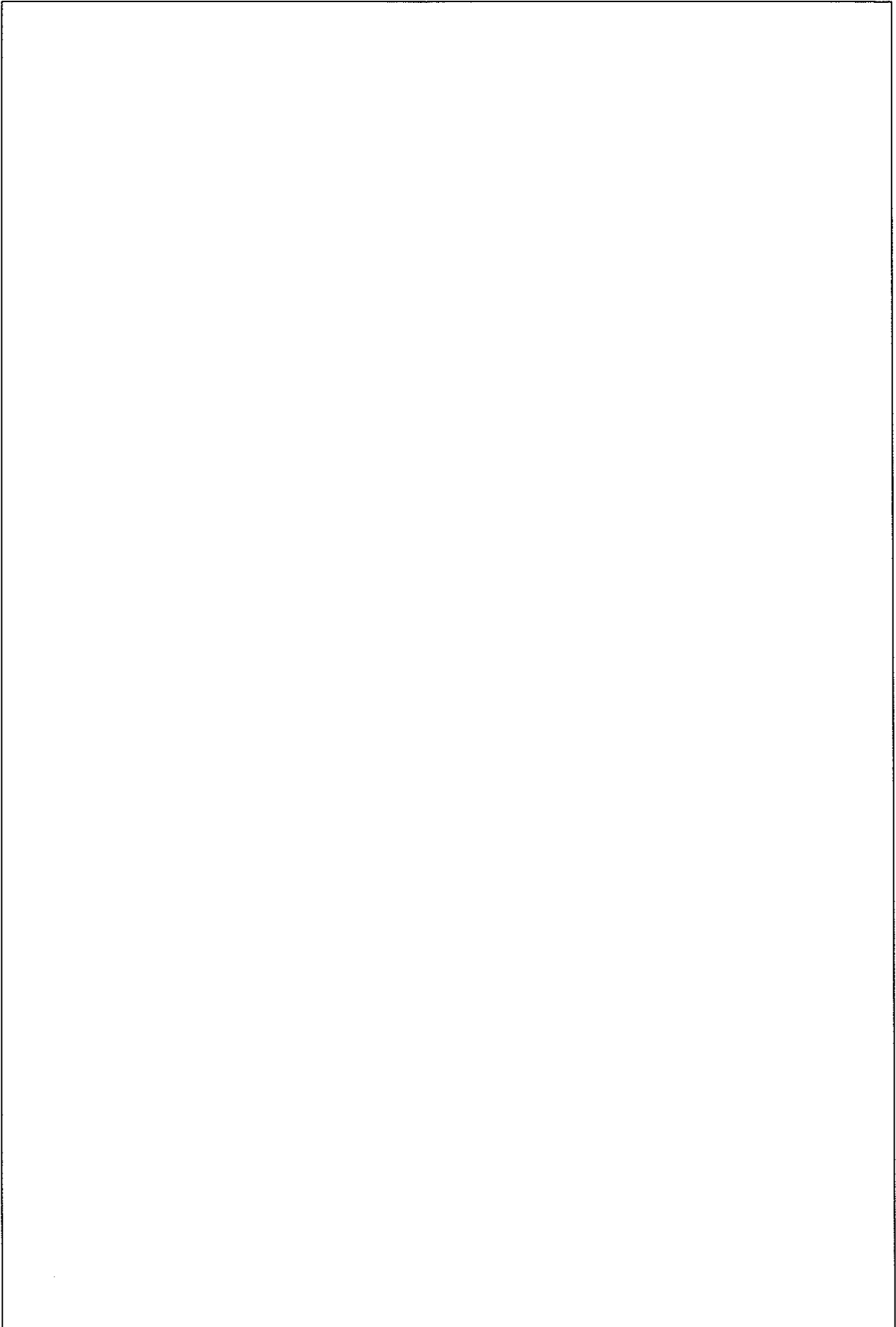
ANSWER:



QUESTION 3 (20 marks)

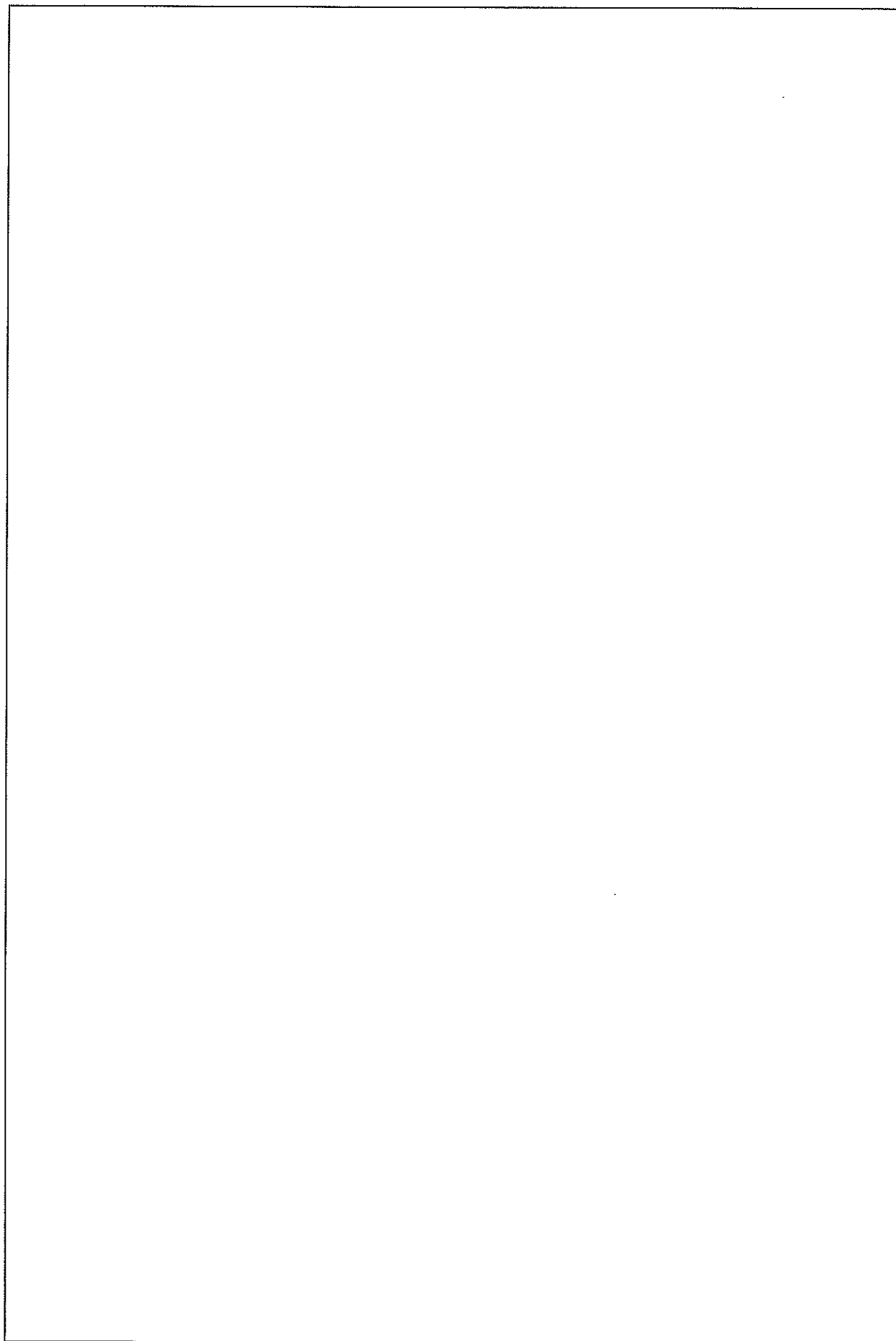
- (3a) Given a NULL-terminated C string **str** (i.e., an array of characters terminated by an element that holds the value 0) that holds a string of at most 8 hexadecimal characters, write a MIPS assembly function “**int convert(str)**” that converts the hexadecimal string into an integer as the function’s return result. The address of **str** is passed as the first argument to **convert**. Note: the ASCII code for ‘0’ to ‘9’ is 48_{10} to 57_{10} , ‘A’ to ‘F’ is 65_{10} to 70_{10} , and ‘a’ to ‘f’ is 97_{10} to 102_{10} , respectively. If any of the characters is not in these ranges, an error status is returned by returning a ‘1’ in $\$v1$. In this case, the content of $\$v0$ does not matter. If there is no error, then $\$v0$ should contain the converted value, and $\$v1$ should be ‘0’. You may use the various branching pseudo-ops for your convenience. (10 marks)

ANSWER:



- (3b) The n factorial is defined as $\text{fac}(n) = n * \text{fac}(n-1)$ with $\text{fac}(1) = 1$, and $n > 0$. Write a MIPS assembly function that computes the n factorial where n is the input argument. (10 marks)

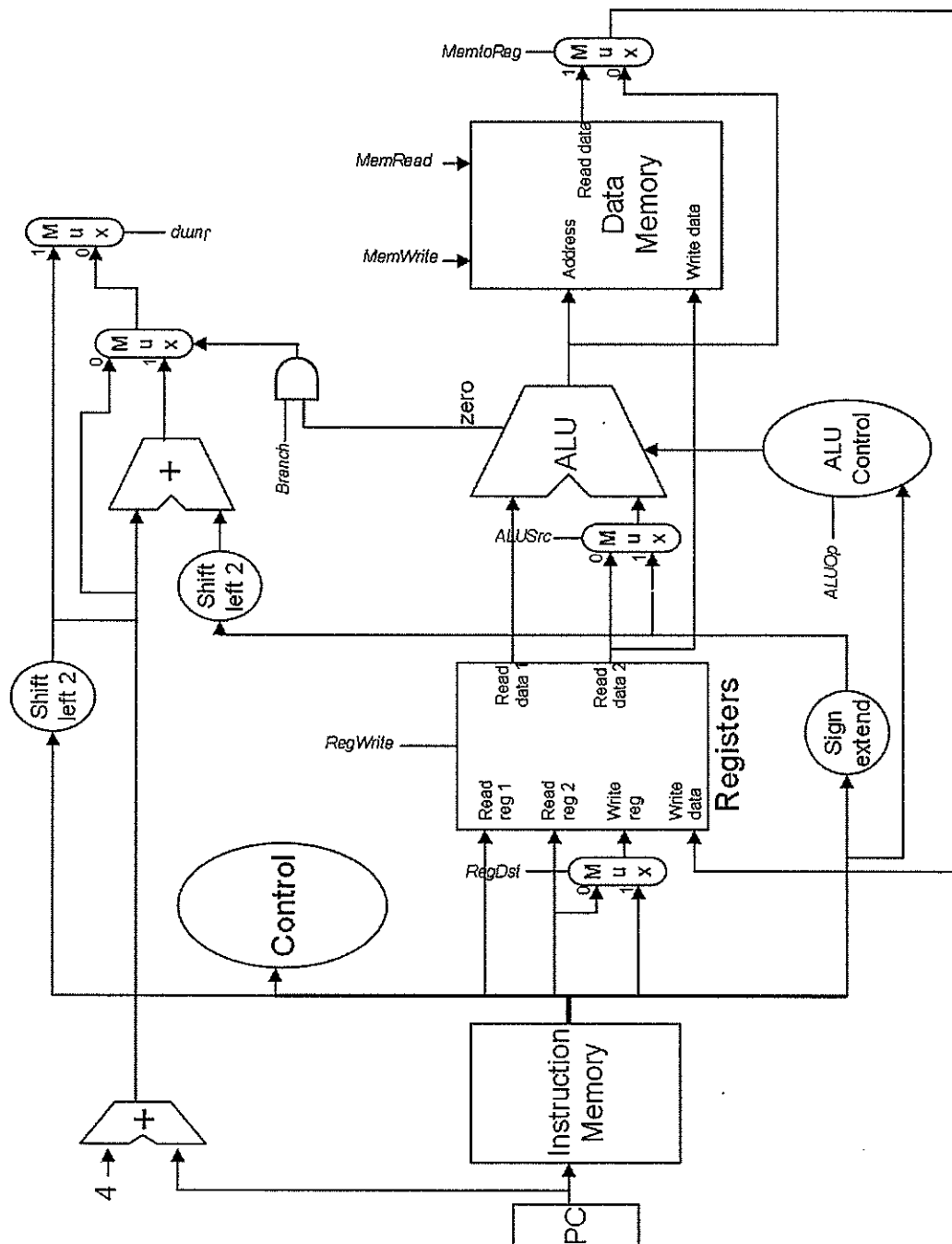
ANSWER:



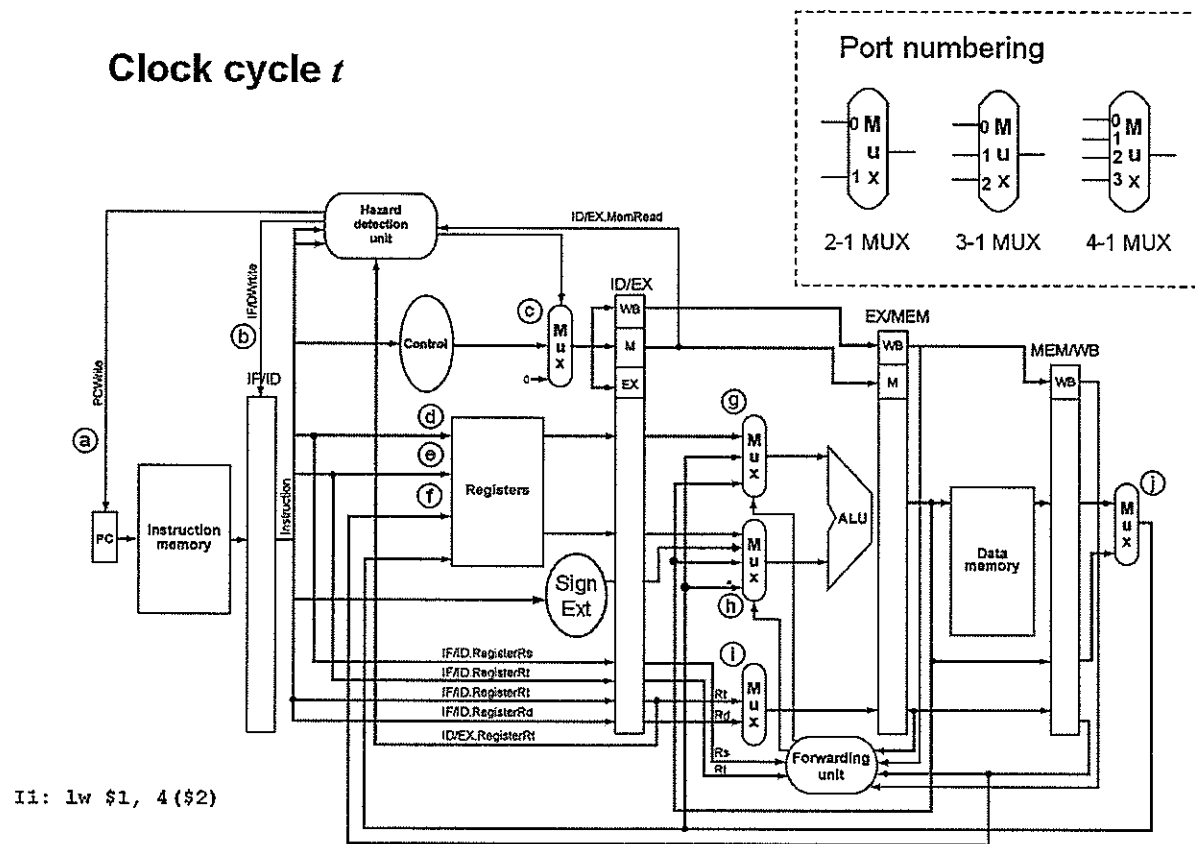
QUESTION 4 (20 marks)

- (4a) The diagram below is a simplified version of the single cycle datapath. By drawing additional components and signal lines, including possibly new control signals (which you should define), show how support for the **jr** ("jump register") instruction can be added. (10 marks)

ANSWER:



- (4b) Consider the following pipelined data path at clock cycle t . (The multiplexor's inputs are labeled as per the diagrams in the box with the dotted border.) The instruction **I1**, namely **lw \$1, 4(\$2)** is currently at the instruction fetch stage.



After **I1**, the following instructions enter the pipeline in order:

I2: add \$1, \$1, \$1
I3: sub \$2, \$1, \$5

I2 starts execution in the IF stage in cycle $t+1$, **I3** starts at cycle $t+2$, and so on. For the cycles, what are the values of the respective signals marked by the circled letters? Assume that the instructions executing/executed prior to cycle t has no effect over the current execution (other than the default $PC = PC+4$). (10 marks)

Matric No:

CS2100 – 2010/11 Sem 1

ANSWER:

Cycle	Signal	Value	Reason
$t+1$	(a)		
	(b)		
	(c)		
	(d)		
	(e)		
	(f)		
$t+2$	(a)		
	(b)		
	(c)		
	(d)		
	(e)		
	(f)		
	(g)		
	(h)		
	(i)		
$t+3$	(a)		
	(b)		
	(c)		
	(d)		
	(e)		
	(f)		
	(g)		
	(h)		
	(i)		

$t+4$	(a)		
	(b)		
	(c)		
	(d)		
	(e)		
	(f)		
	(g)		
	(h)		
	(i)		
	(j)		
$t+5$	(f)		
	(g)		
	(h)		
	(i)		
	(j)		
$t+6$	(f)		
	(j)		

QUESTION 5 (20 marks)

A machine with byte addresses and a word size of 32 bits and address width of 32 bits has a small 1024-block *direct-mapped* write-back cache with each block consisting of 2 words.

- (5a) Propose a C/Java data structure that can be used to hold all the content of the cache. You may assume that the data type “long” holds a 4-byte quantity. (4 marks)

ANSWER:

- (5b)** Using C or Java, write a function that will take a 32 bit address as an argument (typed as “long”) and return a 32 bit data (again, a “long”) if it is found in the cache. It sets a global (integer) flag to indicate a hit (1) or a miss (0). You do not need to concern yourself with further miss processing in the lower cache hierarchy. (8 marks)

ANSWER:

- (5c) Using C or Java, write a code fragment will cause a lot of *compulsory* misses in this cache. (8 marks)

ANSWER:

QUESTION 6 (10 marks)

A machine has a physical address width of 32 bits and a virtual address width of 32 bits. The following content of a full associative 4-entry TLB:

Valid Bit	Virtual Page Number	Physical Page Number	Dirty	Ref	Access
1	0x41081	0x1111	1	20	rw
1	0x6245	0xDEED	0	1	rx
1	0x57C	0x4AC	0	8	rw
0	0x30A2	0x221	1	9	rw

- (6a) Suppose an access to the memory location at virtual address 0x000AF9F4 is translated to the physical address 0x000959F4, what is the size of a page in this machine? (2 marks)

ANSWER:

- (6b) Suppose a malicious program tries to overwrite the text segment of the code by *writing* to the virtual page 0x6245 in the hope that the program will execute it, how would the system respond? (2 marks)

ANSWER:

(6c) A file consists of 8 bytes only. The following shows its content in hexadecimal.

File byte position	Byte content
0	0xD0
1	0xE1
2	0xA2
3	0xD3
4	0xB4
5	0xE5
6	0xE6
7	0xF7

Show how this file would be stored on 5 disk RAID level 3 system employing *odd* parity. “bit[i][j]” refers to “bit j of byte i” (with ‘0’ being the least significant bit/byte). You need to write down the position as well as the value in the answer template. (6 marks)

ANSWER:

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =
bit[][] =	bit[][] =	bit[][] =	bit[][] =	bit =

== END OF PAPER ==