CPSC 313 Sample test 1

Name: ____________________ Student ID: ____________________

Signature: ____________________

- You have 40 minutes to write the 4 questions on this examination.
  A total of 25 marks are available.
- No notes, books or electronic devices are allowed.
- **Justify all of your answers.**
- Keep your answers short. If you run out of space for a question, you have written too much.
- The number in square brackets to the left of the question number indicates the number of marks allocated for that question. Use these to help you determine how much time you should spend on each question.
- Use the back of the pages for your rough work.
- **Good luck!**

Question	Marks
1	
2	
3	
4	
Total	

UNIVERSITY REGULATIONS:

- Each candidate should be prepared to produce, upon request, his/her UBC card.
- No candidate shall be permitted to enter the examination room after the expiration of one half hour, or to leave during the first half hour of the examination.
- CAUTION: candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action.
  1. Having at the place of writing, or making use of, any books, papers or memoranda, electronic equipment, or other memory aid or communication devices, other than those authorised by the examiners.
  2. Speaking or communicating with other candidates.
  3. Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.
- Candidates must not destroy or mutilate any examination material; must hand in all examination papers; and must not take any examination material from the examination room without permission of the invigilator.

[6] 1. The following data has been loaded in memory locations `0x100` and following of the Y86 CPU:

```
0x100:  FF  2A  40  EE  30  F1  02  00
0x108:  00  00  50  71  00  01  00  00
0x110:  00  FF  47  00
```

(recall that our version of the Y86 CPU is a little-endian machine). The program counter is then set to `0x104`, and the program starts executing. What value will register `%edi` contain when the machine halts? You must justify your answer fully.

[3] 2. Saving registers on the stack at the beginning of a function can require multiple `pushl` instructions, and it is error prone since we must use `popl` instructions in the reverse order. It would be nice if our Y86 CPU had `save` and `restore` instructions that save and restore all registers except `%esp` and `%eax` (the latter will contain the return value, so we don't want it restored to its initial value).

Explain briefly why this instruction can not be implemented using our design.

[8] 3. A student wants to implement an instruction `immovl V, (rB)` that stores a constant in the memory location whose address is contained in register `rB`. That, is, the semantics of this instruction is:

$$\texttt{M}_4\texttt{[R[rB]]} \leftarrow V$$

Use the notation described in class (and the book) to document each stage's role in implementing the instruction. That is, use the signal names (i.e., `PC`, `valC`, etc), and the notation `R[x]` for access to register number `x` and $\texttt{M}_\texttt{b}$`[a]` for access to `b` bytes of memory starting at address `a`. List the behaviour of each stage separately:

Fetch:

Decode:

Execute:

Memory:

Write-back:

PC-update:

[8] 4. Complete the following implementation of the `linear_search` function in Y86 assembly language. Hint: my solution uses 7 instructions, three of which are `addl`. You do not need to aim for this minimum, but if you use more than 12 instructions then you are probably making your solution more complicated than necessary. Part marks **will** be given for partially correct answers.

```
#
# Linear search:
# %eax: number of elements of the array.
# %ebx: address of the first element of the array
# %ecx: element we are searching for.
#
# returns %eax: the position of the element, or -1 if not found.
#
linear_search:
    irmovl $1,   %edx  # Decrement the position to search
    subl   %edx, %eax
    jl not_found
```

*****Your Answer goes here*****

```
    ret

not_found:
    irmovl $-1, %eax
    ret
```

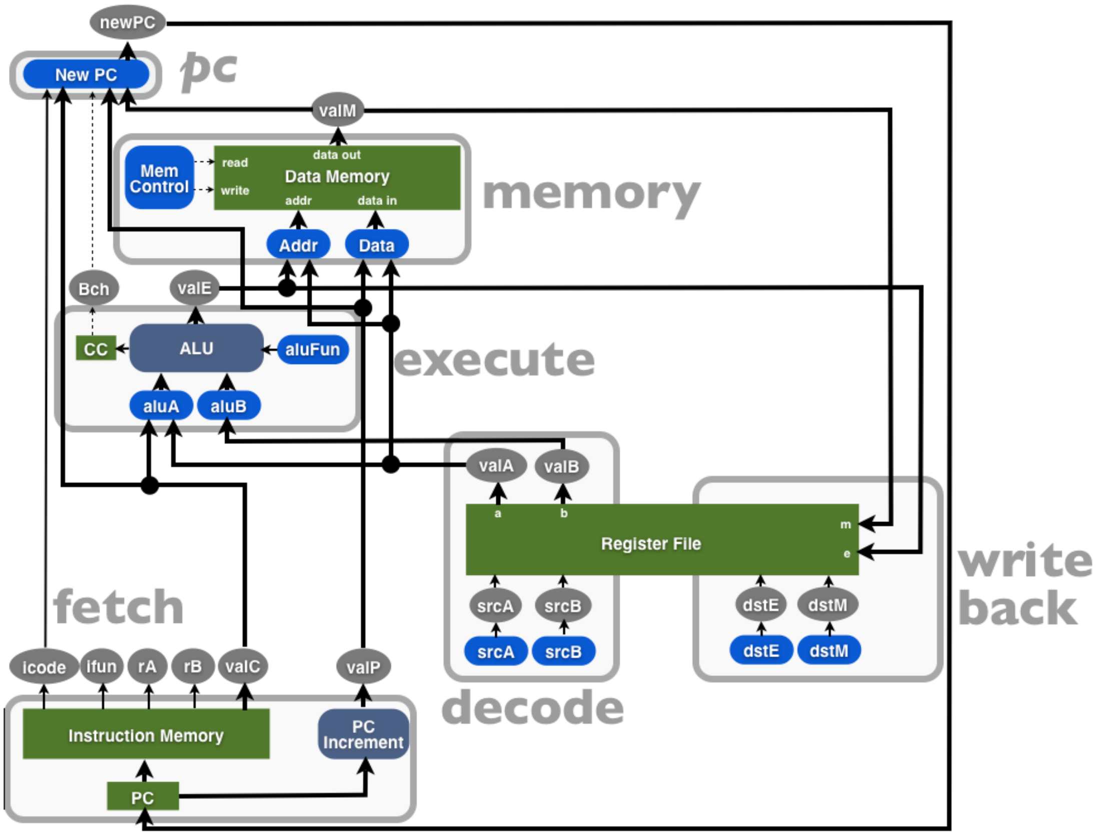
Register Names

0	%eax	%esp	4
1	%ecx	%ebp	5
2	%edx	%esi	6
3	%ebx	%edi	7

Instructions Encoding

Byte	0		1		2	3	4	5
halt	0	0						
nop	1	0						
rrmovl **rA**, **rB**	2	0	**rA**	**rB**				
cmovXX **rA**, **rB**	2	fn	**rA**	**rB**				
irmovl **V**, **rB**	3	0	F	**rB**	**V**			
rmmovl **rA**, **D**(**rB**)	4	0	**rA**	**rB**	**D**			
mrmovl **D**(**rB**), **rA**	5	0	**rA**	**rB**	**D**			
OPl **rA**, **rB**	6	fn	**rA**	**rB**				
jXX **Dest**	7	fn	**Dest**					
call **Dest**	8	0	**Dest**					
ret	9	0						
pushl **rA**	A	0	**rA**	F				
popl **rA**	B	0	**rA**	F				

CPU hardware structure


newPC
New PC
pc
valM
Mem Control
read
write
data out
Data Memory
addr
data in
Addr
Data
memory
Bch
valE
CC
ALU
aluFun
aluA
aluB
execute
valA
valB
a
b
Register File
m
e
srcA
srcB
srcA
srcB
dstE
dstM
dstE
dstM
write back
fetch
icode
ifun
rA
rB
valC
valP
Instruction Memory
PC Increment
PC
decode