Student Name _____ student ID _____

Final examination questions (total 5 pages)
2190250 computer architecture and organization

Three questions, duration 1:30 hour.

This is an "open-book" exam. Students are allowed to bring any textbooks or information on paper with them into the examination room.

Students are NOT allowed to use any calculator, computer, or cell-phone during exam.

Write your answer on this question page.

1) Memory system design

Given a list of 32-bit memory address references, given as word addresses.

0x03, 0xb4, 0x2b, 0x02, 0xbf, 0x58, 0xbe, 0x0e, 0xb5, 0x2c, 0xba, 0xfd

1.1) For each of these references, identify the binary word address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list whether each reference is a hit or a miss, assuming the cache is initially empty.

1.2) For each of these references, identify the binary word address, the tag, the index, and the offset given a direct-mapped cache with 8 two-word blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

Student Name _____ student ID _____

2) Graphic Processing Unit

Inner Product of Matrix multiplication
Two square matrices (n x n) are multiplied together:

c[i][j] = sum (k = 0,n-1) (a[i][[k] * b[k][j])

each term of c[i][j] is called "inner product". when we draw the picture of "inner product" like this

a
```
[ x x x .. ]
[ x ....   ]
[ x ...    ]
```
a[i][k] is the "column" (vertical) of a

b
```
[ x x x... ]
[ x ...    ]
[ x...     ]
```
b[k][j] is the "row" (horizontal) of b

so if we arrange "column" if a, and "row" of b properly (in the memory) we can do the "inner product" of two 4 x 4 matrices with 4-core NPU: Let a column of a is at @100..103, a row of b is at @104..107, use R[2] for partial result, R[0] for a, R[1] for b.

*** Your task is to fill in line 1, 2, 3 (below)

```
ld 0 @104
ld 1 @105
ld 2 @106
ld 3 @107    ; load a from Mem to LDS
ldr 0        ; move LDS to R[0]
ld 0 @108
ld 1 @109
ld 2 @110
ld 3 @111    ; load b from Mem to LDS
ldr 1        ; move LDS to R[1]
mul 2 0 1    ; R[2] = R[0] * R[1]  all cores
```

Now we have partial result in each R[2]. We just add all of them (R[2] of 4 cores) together to have the final result.

Student Name _____ student ID _____

To send one R[2] to another core, we use "broadcast" (bc) instruction. Let move R[2] of core 1 to all R[3]. We then do the same to move R[2] of core 2 to R[4], and R[3] of core 3 to R[5]. Now we have all partial result in R[2], R[3], R[4], R[5], we sum all of them to get the final result. We use R[2] of core 0 to store the result. (We ignore all other core)

```
str 2         ; all R[2] to LDS
bc 3 1        ; LDS[1], R[2] of core 1, to all R[3]

......              (line 1)

......              (line 2)

add 2 2 3
add 2 2 4

......              (line 3)
```

The final result is stored in R[2] of core 0.

**** Your task is the fill in the line 1, 2, 3.

Given

the instruction set of 4-core NPU.

Student Name _____ student ID _____

3) Future of computing

The current supercomputer consists of millions of cores as shown in top500.org list. Many of these systems also include GPU. These super computers consume huge amount of energy. From the perspective of evolution of computer design, what is the future of these machines? Can we increase the size, more core, faster GPU? Where is the limit?

What do you think will be the supercomputer of the next 50 years? What should be the characteristic of the future machine? Extrapolate from the current technology that we use to build computer, what technology should be use for the future computer?

Express your opinion about these questions, give some reason for your idea and some realistic explanation of the technology you mention.

Please write about 200 words for your answer. (that is about a text that is twice as long as this question).

NPU Instruction set (excerpt)

ls 0,1,2,3 local store
r 0..31 register

Data

```
ld ls @ads      LDS[ls] = M[ads]    ;  ls = 0,1,2,3
st ls @ads      M[ads] = LDS[ls]
ldr r           R[r] = LDS          ;  r = 0..31
str r           LDS = R[r]
…
ldw @ads        LDS = M[ads]        ;  load wide
bc r ls         R[r] = LDS[ls]      ;  broadcast
```

ALU operations

```
add r3 r1 r2    R[r3] = R[r1] + R[r2]
sub r3 r1 r2    R[r3] = R[r1] - R[r2]
mul r3 r1 r2    R[r3] = R[r1] * R[r2]
....
lt r3 r1 r2     R[r3] = R[r1] <  R[r2]
le r3 r1 r2     R[r3] = R[r1] <= R[r2]
eq r3 r1 r2     R[r3] = R[r1] == R[r2]
```

Control

```
jmp @ads        pc = ads
jz r @ads       if R[r] == 0, pc = ads
jnz r @ads      if R[r] != 0, pc = ads
```

…

LDS can be named "bus interface" (to join a narrow 32-bit bus, with a wide 32x4 -bit bus to cores) or "broadcast unit" because it can do broadcast one LDS to all cores.

```
bc r ls          LDS[ls] -> all R[r]  ; broadcast
ldw @ads         M[ads] -> all LDS    ; load wide
```

end