COS284 EXAM

Click on a question number to see how your answers were marked and, where available, full solutions.

Question Number
Question 1
Question 2
Question 3
Question 4
Question 5
Question 6
Question 7
Question 8
Question 9
Question 10
Question 11
Question 12
Question 13
Question 14

Performance Summary

Exam Name:	COS284 EXAM
Session ID:	11069341661
Student's Name:	CD (Caleb) Johnstone (c4e68c0161da46538a97731973f1de41)
Exam Start:	Tue Nov 17 2020 08:07:48
Exam Stop:	Tue Nov 17 2020 11:07:46
Time Spent:	2:59:57

Question 1

a)

Given the binary value **10111011**, provide the decimal number represented if the following representations were used (include no white space in your answers):

Unsigned: 187

Sign and magnitude: 59

One's complement: 68

Two's complement: -69

2 marks

b)

Given the following two's complement additions of 4-bit long numbers, state whether overflow has occurred or not:

0011+0111:

● Overflow ○ No Overflow

1000+0111:

○ Overflow ● No Overflow

1 mark

c)

Convert the decimal number **-34.28** to **binary**, making use of the IEEE 754 single precision standard. Provide the following components:

Sign: 1

Exponent: 10000100

Mantissa: 00010010001111010111100001

2 marks

d)

Given the data string **11001** and the polynomial string **101**, calculate the **CRC** remainder and the encoded string:

CRC remainder: 0010

Encoded string: 1100110

3 marks

e)

Given the Hamming encoded data string **00101101**, determine whether the data string is correct or if it is incorrect. Furthermore, determine which bit was incorrectly transmitted. Make use of even parity and assume indexing starts at 0. If the string was correctly transmitted answer with a "?" (without quotation marks) for the incorrect bit.

Correctly transmitted?

● Yes ○ No

Incorrect bit: ?

1 mark

Question 2

For all answers in this question, your variables must be in alphabetical order. For example $b'a$ must be written as ab'.

a)

Simplify the following Boolean expression to a canonical sum of products. (Hint: A Karnaugh-map may be useful.)

(If the number of products in your sum is less than the number of spaces provided for your answer, just fill the remaining space(s) with a 0. For example, if your answer is $f(x,y,z) = x + z'$ and three spaces have been provided, put a 0 in the final space, effectively changing your answer to the (still algebraically correct $f(x,y,z) = x + z' + 0$, even though this latter expression does not conform to the rules for canonical Boolean expressions.)

Simplify:

$f(a,b,c) = bc + a'b + b'c + a'b'$

$f(a,b,c) =$ a' + c + 0

3 marks

b)

Simplify the following Boolean expression to a canonical sum of products. (Hint: A Karnaugh-map may be useful.)

If the spaces provided for answers exceed the number of products in your sum, just fill the remaining space(s) with a 0. For example, if your answer is $f(x,y,z) = x + z'$ and three spaces have been provided, put a 0 in the final space, effectively changing your answer to the (still algebraically correct $f(x,y,z) = x + z' + 0$, even though this latter expression does not conform to the rules for canonical Boolean expressions).

Let $p(a,b,c,d) = 0$ if the parity of (a,b,c,d) is even - that is if none, two or all four of the values of a, b, c and d are a 1. Otherwise the parity is odd and $p(a,b,c,d) = 1$.

Now consider $f(a,b,c,d) = (a+b)(c+d)$ if $p(a,b,c,d) = 0$. If $p(a,b,c,d) = 1$, we do not care what the value of $f(a,b,c,d)$ is.

In the simplified form (as specified above):

$f(a,b,c,d) =$ ac + ad + bc + bd .

4 marks

c)

Consider an edge triggered JK flipflop. It triggers on the rising edge of a clock pulse. For each of the sets of inputs below, indicate its state moments after the rising edge of the clock pulse has occurred. For each case assume that its state prior to the clock pulse was q.

1. J=0, K=0; the new state is q
2. J=1, K=1; the new state is q'

1 mark

d)

Consider a level triggered JK flipflop. It triggers on a high clock pulse. For each of the sets of inputs below, indicate its state moments after it has become 1. For each case assume that its state prior to the clock pulse was q.

1. J=1, K=0; the new state is q
2. J=0, K=1; the new state is q'

1 mark

Question 3

The opcodes for the basic MARIE instructions are:

0001 LOAD X

0010 STORE X

0011 ADD X

0100 SUBT X

0101 INPUT

0110 OUTPUT

0111 HALT

1000 SKIPCOND

1001 JUMP X

The condition codes for SKIPCOND are

- 00 for AC < 0
- 01 for AC = 0
- 11 for AC > 0

The instructions available in MARE microcode (expressed in RTL) are the following:

0) **NOP**
1) **AC** ← **0**
2) **AC** ← **MBR**
3) **AC** ← **AC - MBR**
4) **AC** ← **AC + MBR**
5) **AC** ← **InReg**
6) **IR** ← **M[MAR]**
7) **M[MAR]** ← **MBR**
8) **MAR** ← **IR[11-0]**
9) **MAR** ← **MBR**
10) **MAR** ← **PC**
11) **MBR** ← **M[MAR]**
12) **MBR** ← **AC**
13) **MBR** ← **M[IR]**

14) **OutReg ← AC**
15) **PC ← IR[11-0]**
16) **PC ← MBR**
17) **PC ← PC + 1**
18) **PC ← PC + 1 if AC = 0**
19) **PC ← PC + 1 if AC > 0**
20) **PC ← PC + 1 if AC < 0**

The NOP instruction is a no-operation instruction, that does nothing. You are only allowed to use it at the end of microcode programs (to pad your code to fill the provided answer space).

a)

Consider the following MARIE program (starting at address 100 hexadecimal)

```
LOOP:  INPUT
       ADD SUM
       STORE SUM
       LOAD COUNT
       SUBT ONE
       SKIPCOND (AC = 0)
       JUMP LOOP
DONE: HALT
COUNT DEC 10
ONE    DEC 1
SUM    DEC 0
```

'Assemble' the program by converting each assembly instruction into a machine code instruction expressed as a four-digit hexadecimal number. Only the first seven machine code instructions have to be included in your answer. You may use upper and/or lower case to represent hexadecimal numbers; however only write down the four hexadecimal digits of each machine code instruction - you do not have to indicate that those values are expressed as hexadecimal values. Fill any unused bits in an instruction with zeroes.

100 5000

101	310A
102	210A
103	1108
104	4109
105	8001
106	9100
107	7000

4 marks

b)

Write a MARIE program fragment that will implement the STORE instruction (after the instruction has been fetched and decoded). Use the RTL instructions provided at the start of this question. Convert the instructions to micro-opcodes and provide the (decimal) micro-opcodes below.

8

12

7

3 marks

Question 4

a)

Assume that some computer uses byte addressable memory and uses 8-bit memory addresses. The following table represents the entire contents of its memory. The value at address A9 occurs where row A_ intersects with column _9; it happens to be 33 (hexadecimal, like everything else in this question).

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E
0_	92	2e	d8	7f	0f	d6	3c	04	9a	75	7a	52	aa	05	d1
1_	af	e5	74	9d	9e	96	25	da	66	b5	2c	e9	48	72	b9
2_	6a	e5	aa	2	49	50	42	9a	24	d8	00	69	99	e7	c7
3_	cf	bc	c6	3c	d9	c4	75	ef	d8	0	28	c9	69	57	56
4_	0f	b8	ba	b3	2c	89	d8	d8	97	d8	71	fd	22	a2	ca
5_	cf	79	5e	ec	ed	f7	5f	f4	f7	1f	fc	d3	df	e4	13
6_	79	f8	fc	e9	d3	f8	9f	e2	38	7e	fc	f4	49	f0	bf
7_	79	18	ef	3f	7e	f4	64	ff	f1	f3	27	8f	1e	3f	8f
8_	fd	87	8f	9f	3f	fa	a7	f8	e9	df	66	3a	e1	67	5d
9_	d2	f2	3b	8c	23	eb	70	ff	fb	0f	f2	a9	82	f3	9f
A_	57	e9	ec	bb	53	c1	d7	9d	ff	33	38	ff	fd	27	4f
B_	76	e7	ff	6b	7c	da	ce	3f	fd	32	9f	95	cb	f1	de
C_	1f	67	8b	71	d6	5f	7d	59	7d	d3	3b	f6	1f	3e	7c
D_	e4	c9	d6	f3	7f	f4	f0	e9	a3	f0	fc	1f	ed	3f	79
E_	ec	9f	e2	87	df	69	8d	b7	7e	f4	fc	7f	ce	16	9f
F_	45	b5	f5	7b	7f	19	17	65	36	cf	26	79	9a	17	93

The computer has only one index register, and its value happens to be 10 at the moment. The computer also has one base register, and it contains the value 50 at the moment.

Assume that the computer described above executes the instruction

ADD 44

(which adds a byte to the accumulator)

What value will be added to the accumulator if the ADD instruction above uses the following addressing mode?

1. Immediate 44
2. Direct 2c
3. Indirect 99
4. Indexed 2c
5. Based 23

5 marks

b)

Rewrite the following expression in postfix notation. (Normal priority rules apply)

|A+B*C|

In this equation |x| indicates the absolute value of x. Assume the postfix operator to determine absolute value is the word **abs**.

A	B	C	*	+	abs

3 marks

Question 5

a)

In all the cases in this part assume that the word addressable computer uses 32-bit addresses. For each scenario input how many bits are used for the fields indicated.

1. The computer uses direct mapped caching with 1 kiloword per block and space is provided for 16 blocks in the cache. How many bits are allocated to the following fields in each address?

1. **Tag** 18

2. **Offset** 10

2. The computer uses fully associative caching with 2 kilowords per block; the cache consists of 16 blocks How many bits are allocated to the following fields in each address?

1. **Tag** 20

2. **Offset** 12

3. The computer uses 4-way set associative mapping with 1 kiloword per block. Each cache set may contain up to 4 blocks. How many bits are allocated to the following fields in each address?

1. **Tag** 18
2. **Set** 4

3 marks

b)

Assume a computer uses 64 kilobytes of byte addressable memory, with 16 bytes per block. It provides 8 blocks of cache memory. It uses 4-way set associative caching and a FIFO approach for block replacement.

Prior to the memory accesses listed below, its cache is empty.

It then access the following (hexadecimal) memory addresses in sequence

1. 1234
2. B055
3. BA5E
4. BEE5
5. fa11
6. F00D
7. FADE
8. f1ee
9. FEED
10. FEE5
11. FACE
12. F00D

Note that all the values above are addresses of bytes in memory. In the questions that follow, the answer will sometimes be a block number that - in this case - will always consist of three hexadecimal digits. (Note that the answer will in some cases NOT be a block number.)

For any memory access the following may happen:

- The value to be accessed may be in cache. In this case, write "Hit" (without the quotation marks) in the gap provided.

- The value may not be in cache, but can be loaded into an unused cache block. In this case, write "Open" (without the quotation marks) in the gap provided.
- The value may not be in the cache and it may be necessary to evict a block. In this case, write the number of the memory block to be evicted in the gap provided. Provide your answer in hexadecimal.

You may user upper-, lower or mixed case for your answers.

Note that you will have to determine all 12 answers, but only your answers for accesses 5 to 12 are tested below.

Ensure that you enter digits where required; do NOT enter the letter O where the digit 0 would be expected.

Provide the answers requested above for this computer.

1. 1234
2. B055
3. BA5E
4. BEE5
5. fa11 Open
6. F00D Open
7. FADE 123
8. f1ee Open
9. FEED Open
10. FEE5 Hit
11. FACE BEE
12. F00D Hit

8 marks

Question 6

a)

Given some arbitrary system, we have a choice of upgrading either the CPU or the Hard Drive. The CPU upgrade will improve CPU performance by 40% while the Hard Drive upgrade will improve Hard Drive performance by 65%. Processes tend to spend 70% of their time in the processor, and 30% of their time awaiting data from disk. The CPU upgrade will cost R 2000 and the Hard Drive upgrade will cost R 800. Round all answeers with a fractional part to two decimal places.

Calculate:

The percentage speedup if the CPU is upgraded: 25 %

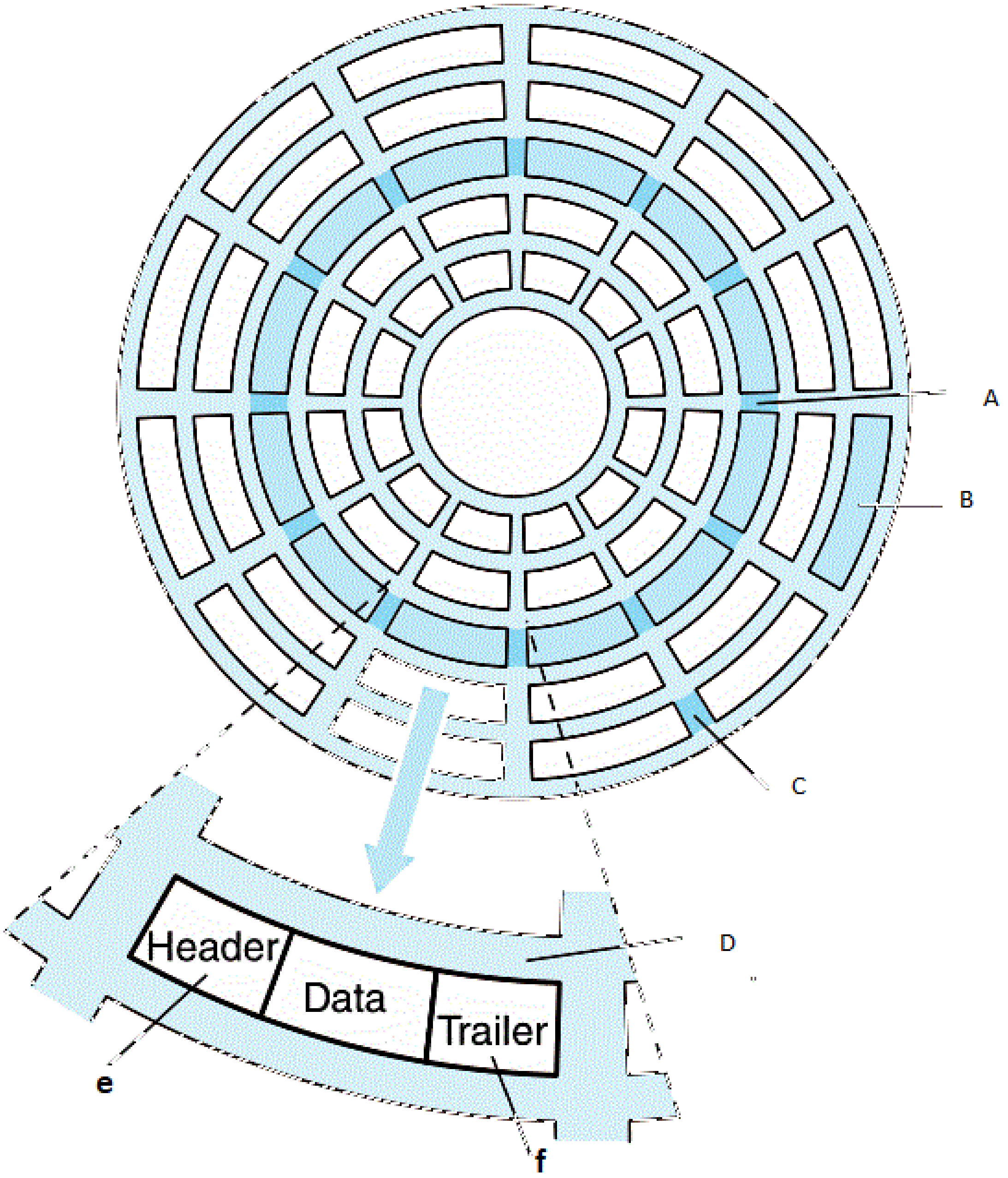
The percentage speedup if the Hard Drive is updgraded: 13.40 %

Select which component should be upgraded to maximise performance:

● CPU ○ Hard Drive

1.5 marks

b)



Consider the provided image of a disk. Provide the letter from the figure that best fits each term below:

Sector: B

Track: A

Intertrack gap: D

Intersector gap: C

During data access, which two components are accessed by the read/write head? Provide your answer with no spaces as a comma seperated list, in the order they are accessed during data access. cylinder,sector

3.5 marks

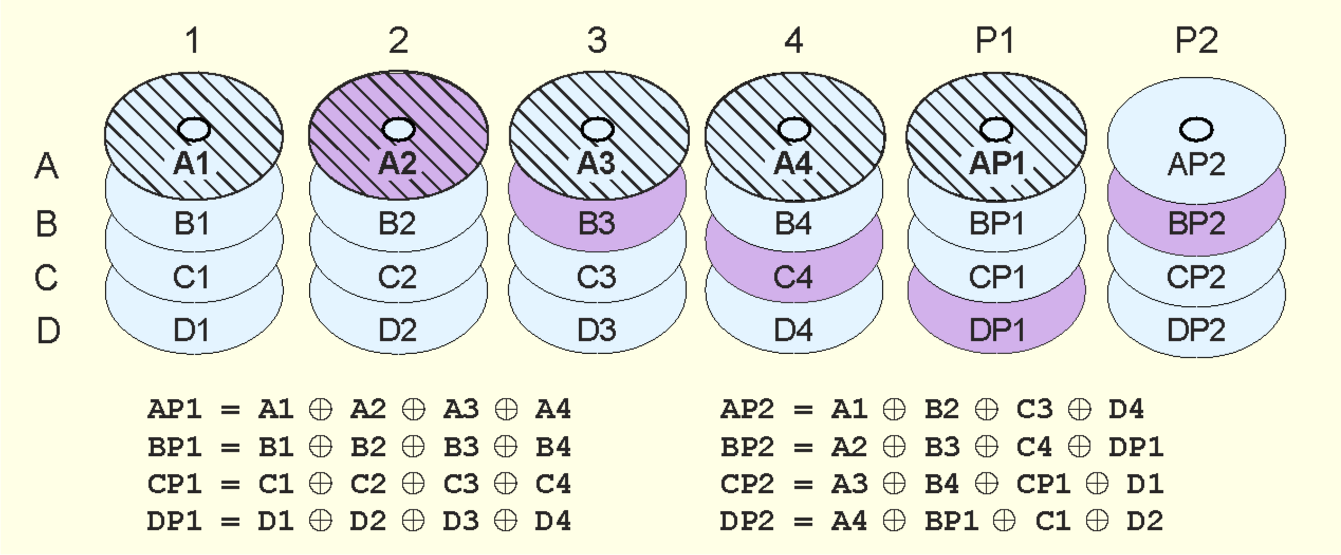
c)

Provide the acronyms (only, and case does not matter) for the two **most** important reliability metrics for SSD as per **JEDEC**.

UBER and TBW .

1 mark

d)



$AP1 = A1 \oplus A2 \oplus A3 \oplus A4$
$BP1 = B1 \oplus B2 \oplus B3 \oplus B4$
$CP1 = C1 \oplus C2 \oplus C3 \oplus C4$
$DP1 = D1 \oplus D2 \oplus D3 \oplus D4$

$AP2 = A1 \oplus B2 \oplus C3 \oplus D4$
$BP2 = A2 \oplus B3 \oplus C4 \oplus DP1$
$CP2 = A3 \oplus B4 \oplus CP1 \oplus D1$
$DP2 = A4 \oplus BP1 \oplus C1 \oplus D2$

Given the above image depicting **RAID DP**, answer the following questions:

Assuming that disk **2** and **P1** are lost by some means, provide the right side of the equation to recalculate the lost data segment. Include no spaces in your answer. There are no partial marks for this question.

B2= A1 **XOR** C3 **XOR** D4 **XOR** AP2

2 marks

e)

Use the figure and disk losses from part d). Assuming that disk **3** and **P2** are lost by some means, provide the right side of the equation to recalculate the lost data segment. Include no spaces in your answer. There are no partial marks for this question.

D3= D1 **XOR** D2 **XOR** D3 **XOR** DP1

2 marks

Question 7

a)

Consider the following features. In each case say whether the statement is primarily associated with RISC or CISC architectures.

1. Complexity in microcode.

○ RISC ● CISC

2. Many addressing modes.

○ RISC ● CISC

3. Many instructions can access memory.

○ RISC ● CISC

4. Multiple register sets.

● RISC ○ CISC

5. Parameter passing through register windows.

● RISC ○ CISC

6. Single-cycle instructions.

● RISC ○ CISC

7. Hardwired control

● RISC ○ CISC

8. Variable length instructions.

○ RISC ● CISC

4 marks

b)

Provide the 4-letter acronym assigned by Flynn's taxonomy to each of the following types of processors:

1. Vector processor SIMD
2. Uniprocessor SISD
3. Multimprocessor MIMD

3 marks

c)

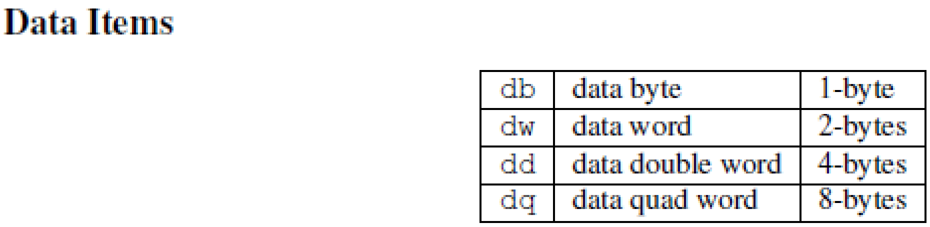
Consider a 6-dimensional hypercube.

1. What is the maximum number of other nodes any given node may be connected to? 6
2. What is the longest path from one node to another node in the hypercube? 6
3. How many processors does the hypercube accommodate? 64

3 marks

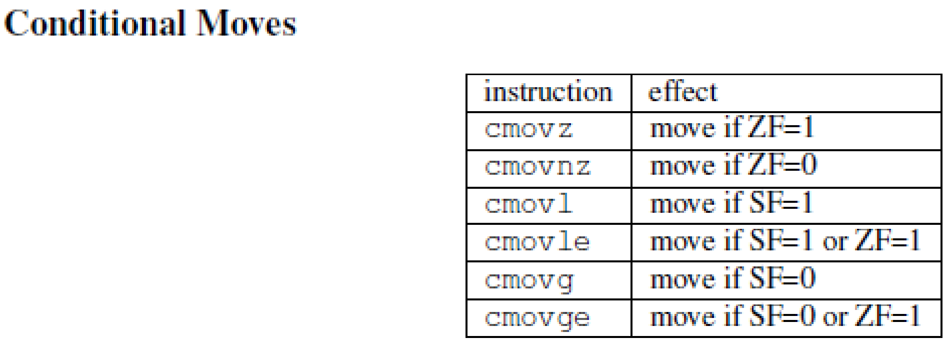
Question 8

The following tables are provided as a reference that may be used if needed, for all ASM questions.



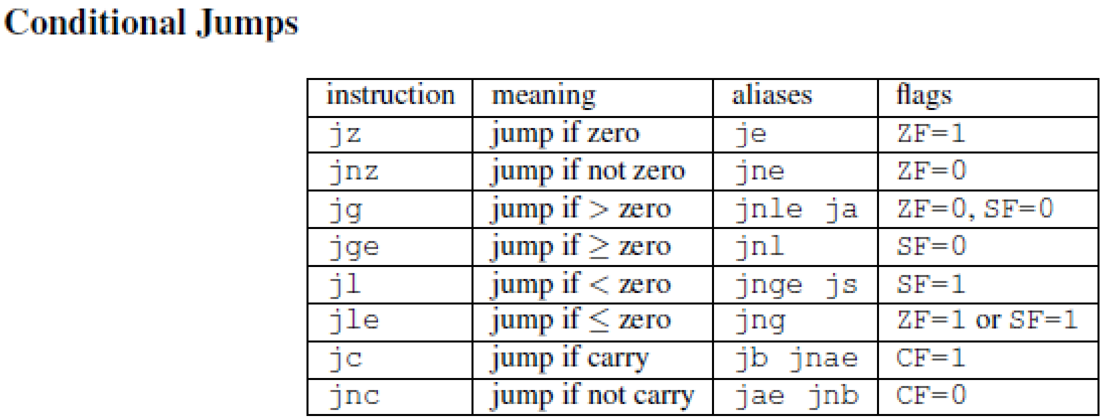
Data Items

db	data byte	1-byte
dw	data word	2-bytes
dd	data double word	4-bytes
dq	data quad word	8-bytes



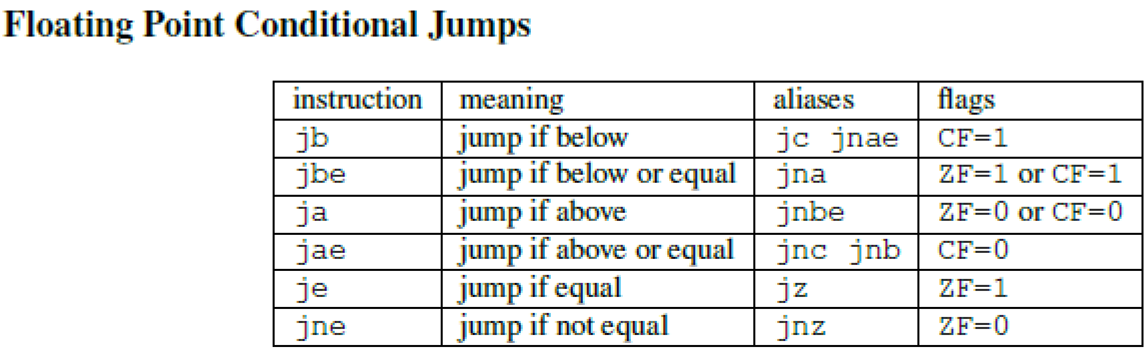
Conditional Moves

instruction	effect
cmovz	move if ZF=1
cmovnz	move if ZF=0
cmovl	move if SF=1
cmovle	move if SF=1 or ZF=1
cmovg	move if SF=0
cmovge	move if SF=0 or ZF=1



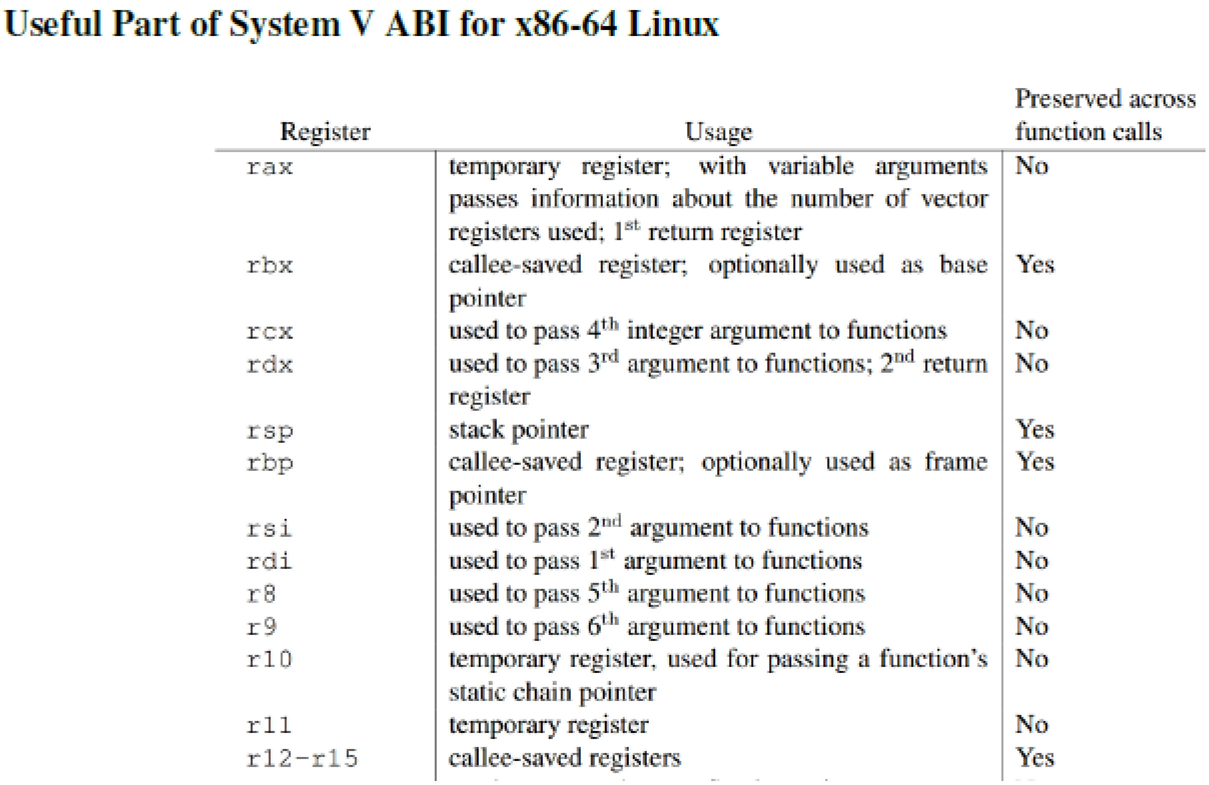
Conditional Jumps

instruction	meaning	aliases	flags
jz	jump if zero	je	ZF=1
jnz	jump if not zero	jne	ZF=0
jg	jump if > zero	jnle ja	ZF=0, SF=0
jge	jump if ≥ zero	jnl	SF=0
jl	jump if < zero	jnge js	SF=1
jle	jump if ≤ zero	jng	ZF=1 or SF=1
jc	jump if carry	jb jnae	CF=1
jnc	jump if not carry	jae jnb	CF=0



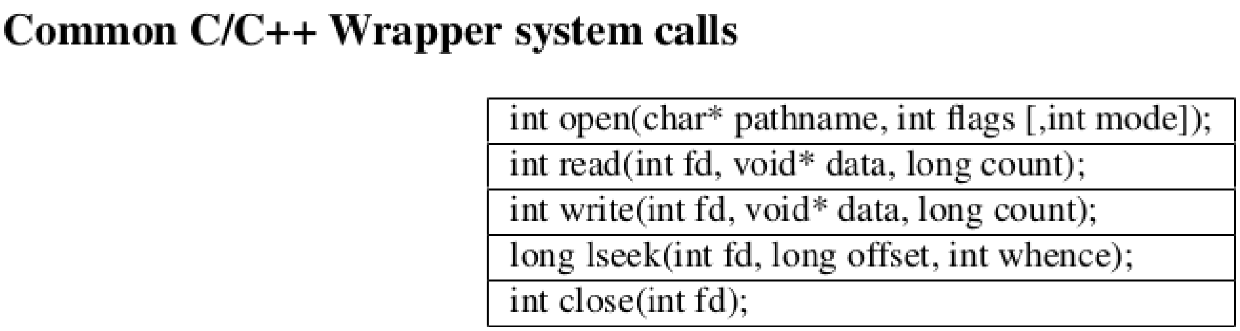
Floating Point Conditional Jumps

instruction	meaning	aliases	flags
jb	jump if below	jc jnae	CF=1
jbe	jump if below or equal	jna	ZF=1 or CF=1
ja	jump if above	jnbe	ZF=0 or CF=0
jae	jump if above or equal	jnc jnb	CF=0
je	jump if equal	jz	ZF=1
jne	jump if not equal	jnz	ZF=0



Useful Part of System V ABI for x86-64 Linux

Register	Usage	Preserved across function calls
rax	temporary register; with variable arguments passes information about the number of vector registers used; 1st return register	No
rbx	callee-saved register; optionally used as base pointer	Yes
rcx	used to pass 4th integer argument to functions	No
rdx	used to pass 3rd argument to functions; 2nd return register	No
rsp	stack pointer	Yes
rbp	callee-saved register; optionally used as frame pointer	Yes
rsi	used to pass 2nd argument to functions	No
rdi	used to pass 1st argument to functions	No
r8	used to pass 5th argument to functions	No
r9	used to pass 6th argument to functions	No
r10	temporary register, used for passing a function's static chain pointer	No
r11	temporary register	No
r12-r15	callee-saved registers	Yes



Common C/C++ Wrapper system calls

int open(char* pathname, int flags [,int mode]);
int read(int fd, void* data, long count);
int write(int fd, void* data, long count);
long lseek(int fd, long offset, int whence);
int close(int fd);

Question 9

Assume that register **rax** stores the following hex string:

3123456789ABCDEF

a)

What hex hexadecimal should stored in each of the following registers: (just type the hexadecimal value, do not add a 0x prefix, do not include spaces)

eax: 89ABCDEF

ax: CDEF

al: EF

ah: CD

2 marks

b)

Assume the following assembler instructions were executed:

mov rbx, 0xfff
shl rbx, 12
not rbx
and rax, rbx

What hexadecimal value be stored in **eax**? (just type the hexadecimal value, do not add a 0x prefix, do not include spaces)

Answer: DEF

2 marks

c)

Assume the following assembler instructions were executed:

```
mov rbx, 0xfff
shl rbx, 12
neg rbx
and rax, rbx
```

What hexadecimal value be stored in **eax**? (just type the hexadecimal value, do not add a 0x prefix, do not include spaces)

Answer: 89

2 marks

Question 10

Complete the following ASM functions that return the smallest of three given parameters (you may assume they are
distinct).

a)

```
;long smallestL(long a, long b, long c)
smallestL:
  mov rax, (I)
  (II) rdi,rsi
  (III) rax,rsi
  (IV) rax,rdx
  (V) rax,rdx
  ret
```

Answers:

(I) rdi

(II) cmp

(III) cmovl

(IV) cmp

(V) cmovl

2.5 marks

b)

```
;float smallestF(float a, float b, float c)
smallestL:
  (I) xmm0,xmm1
  (II) label1
  (III) xmm0,xmm1
  label1:
  (IV) xmm0,xmm2
  (V) label2
  (VI) xmm0,xmm2
  label2:
  ret
```

Answers

(I) ucomiss

(II) ja

(III) movss

(IV) ucomiss

(V) ja

(VI) movss

3 marks

Question 11

Complete the implementation of the ASM function that determines if every component of **array1** is **strictly**
larger than the corresponding component of **array2**. If this condition is met the function must return **1** and if
not a **0**. For simplicity, assume that the arrays are of equal size and the size will always be positive.

```
;long strictly_larger(double* array1, double* array2, long size)
strictly_larger:
 xor rcx,rcx
 mov rax,1
 while:
   cmp rcx,(a)
   je done
   (b) xmm0, (c)
   (d) xmm1, (e)
   inc rcx
   (f) xmm0, xmm1
   (g) while
   mov rax, 0
  done:
  ret
```

(a) rdx

(b) movsd

(c) [rdi+8*rcx]

(d) movsd

(e) [rsi+8*rcx]

(f) ucomisd

(g) je

Question 12

Consider the following C/C++ struct.

```
struct EX
{
   int a;
   \\location 1
   char b[2];
   \\location 2
   long c;
   \\location 3
   char d[3];
   \\location 4
};
```

a)

Where and how much padding would be added to **EX** by C/C++ to preserve alignment conditions? Indicate the location number and number of bytes. Your answer must list the lower location number first. If an additional location is not needed answer ? for both the location and the number of bytes, this should be the last answer if it is needed.

Padding:

Location 2 Number of Bytes 2

Location 4 Number of Bytes 1

Location ? Number of Bytes ?

5 marks

b)

What would the the **sizeof** function return when called on an instance of **EX**?

Answer: 24

1 mark

Question 13

Assume we have implemented a binary tree in ASM, where each node contains a long. In our tree the left child’s value is smaller than the
current node’s value which is smaller than the right node’s value. Each node is represented by

```
struc node
.value resq 1
.left_child resq 1
.right_child resq 1
endstruc
```

Each node was allocated using the C malloc function.

a)

Complete an assembler function with the following prototype to deallocate a binary tree. You must always deallocated the left subtree before the right subtree. Empty nodes are represented as the NULL pointer (0). Assume that all needed C functions have been included. *hint: the method is recursive*

```
; void deallocate(node* root)
deallocate:
  push rbp
  mov rbp,rsp
  sub rsp, 16
  .current_node equ 0
  mov (I), rdi
  cmp (II)
  je .end
  (III)
  call deallocate
  mov rdi, (IV)
  (V)
  call deallocate
```

```
(VI)
call free
.end:
(VII)
ret
```

Answers:

(I) [rsp+.current_node]

(II) rdi, 0

(III) mov rdi, [rsp+.current_node+node.left_child]

(IV) [rsp+.current_node]

(V) mov rdi, [rdi+node.right_child]

(VI) mov rdi, [rsp+.current_node]

(VII) leave

6.5 marks

b)

Complete an assembler function with the following prototype to return the pointer to the node with the largest value. Assume the tree is not empty.

```
; node* find_largest(node* head)
find_largest:
push rbp
mov rbp, rsp
next_layer:
mov rax, (I)
mov rdi, (II)
cmp rdi, (III)
(IV) next_layer
.done
leave
ret
```

(I) [rdi+node.value]

(II) [rdi+node.right_child]

(III) 0

(IV) jne

3.5 marks

Question 14

For this question assume each square matrix was allocated as follows:

```
long** matrix;
matrix = malloc(N_ELEMENTS * sizeof(long*));
for (int i = 0; i < N_ELEMENTS; i++)
  matrix[i] = malloc(N_ELEMENTS * sizeof(long));
```

With this in mind consider the following C function that calculates the product **AB** and stores the result in a pre-allocated matrix **C**. All matrices are square.

```
void product(long** A, long** B, long** C, long size)
{
  for(long j=0;j<size;++j)
    for(long k=0;k<size;++j)
    {
       r=B[k][j];
       for(long i=0;i<size;i++)
         C[i][j]+=A[i][K]*r;
     }

  return;
}
```

In the most inner loop iteration the chances of a cache miss on

Matrix A is:

○ No risk ○ Low risk ● High risk

In the most inner loop iteration the chances of a cache miss on

Matrix B is:

● No risk ○ Low risk ○ High risk

In the most inner loop iteration the chances of a cache miss on

Matrix C is:

○ No risk ○ Low risk ● High risk

Created using Numbas (https://www.numbas.org.uk), developed by Newcastle University (http://www.newcastle.ac.uk).