UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Sekolah Pendidikan Profesional dan Pendidikan Berterusan (SPACE)

FINAL EXAMINATION / *PEPERIKSAAN AKHIR*
SEMESTER II - SESSION 2023 / 2024
PROGRAM KERJASAMA

COURSE CODE *KOD KURSUS*	:	DDWD 1223
COURSE NAME *NAMA KURSUS*	:	COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE *ORGANISASI KOMPUTER DAN BAHASA HIMPUNAN*
YEAR / PROGRAMME *TAHUN / PROGRAM*	:	1 DDWD
DURATION *TEMPOH*	:	2 HOURS 30 MINUTES / *2 JAM 30 MINIT*
DATE *TARIKH*	:	MAY 2024 / JUNE 2024 *MEI 2024 / JUN 2024*

INSTRUCTION / *ARAHAN:*

1. The question paper consists of **3 PARTS**: A, B and C.
   *Kertas soalan terdiri daripada **3 BAHAGIAN**: A, B dan C.*

2. Answer **ALL** questions and write your answers in this paper.
   *Jawab **SEMUA** soalan dan tulis jawapan anda di dalam kertas ini.*

(You are required to write your name and your college's name on your answer script)
(Pelajar dikehendaki tuliskan nama dan nama kolej pada skrip jawapan)

NAME / *NAMA PELAJAR*	:	.....
I.C NO. / *NO. K/PENGENALAN*	:	.....
YEAR / PROGRAMME *TAHUN / PROGRAM*	:	.....
COLLEGE'S NAME *NAMA KOLEJ*	:	.....

This examination paper consists of **28** pages including the cover.
*Kertas soalan ini mengandungi **28** muka surat termasuk kulit hadapan.*

UTM UNIVERSITI TEKNOLOGI MALAYSIA | School of Professional and Continuing Education (SPACE)

PUSAT PRGORAM KERJASAMA

PETIKAN DARIPADA PERATURAN AKADEMIK
ARAHAN AM – PENYELEWENGAN AKADEMIK

1. SALAH LAKU SEMASA PEPERIKSAAN

1.1. Pelajar tidak boleh melakukan mana-mana salah laku peperiksaan seperti berikut :-

1.1.1. memberi dan/atau menerima dan/atau memiliki sebarang maklumat dalam bentuk elektronik, bercetak atau apa jua bentuk lain yang tidak dibenarkan semasa berlangsungnya peperiksaan sama ada di dalam atau di luar Dewan/Bilik Peperiksaan melainkan dengan kebenaran Ketua Pengawas; atau

1.1.2. menggunakan maklumat yang diperoleh seperti di atas bagi tujuan menjawab soalan peperiksaan; atau

1.1.3. menipu atau cuba untuk menipu atau berkelakuan mengikut cara yang boleh ditafsirkan sebagai menipu semasa berlangsungnya peperiksaan; atau

1.1.4. lain-lain salah laku yang ditetapkan oleh Universiti (seperti membuat bising, mengganggu pelajar lain, mengganggu Pengawas menjalankan tugasnya).

2. HUKUMAN SALAH LAKU PEPERIKSAAN

2.1. Sekiranya pelajar didapati telah melakukan pelanggaran mana-mana peraturan peperiksaan ini, setelah diperakukan oleh Jawatankuasa Peperiksaan Fakulti dan disabitkan kesalahannya, Senat boleh mengambil tindakan dari mana-mana satu yang berikut :-

2.1.1. memberi markah SIFAR (0) bagi keseluruhan keputusan peperiksaan kursus yang berkenaan (termasuk kerja kursus); atau

2.1.2. memberi markah SIFAR (0) bagi semua kursus yang didaftarkan pada semester tersebut.

2.2. Jawatankuasa Akademik Fakulti boleh mencadangkan untuk diambil tindakan tatatertib mengikut peruntukan Akta Universiti dan Kolej Universiti, 1971, Kaedah-kaedah Universiti Teknologi Malaysia (Tatatertib Pelajar-pelajar), 1999 bergantung kepada tahap kesalahan yang dilakukan oleh pelajar.

2.3. Pelajar yang didapati melakukan kesalahan kali kedua akan diambil tindakan seperti di perkara dan dicadang untuk diambil tindakan tatatertib mengikut peruntukan Akta Universiti dan Kolej Universiti, 1971, Kaedah-kaedah Universiti Teknologi Malaysia (Tatatertib Pelajar-pelajar), 1999.

SECTION A / *BAHAGIAN A*

10 MARKS / *10 MARKAH*

TRUE AND FALSE QUESTIONS / *SOALAN SALAH DAN BETUL*

Answer all questions. / *Jawab semua soalan.*

NO. *BIL*	STATEMENT *PERNYATAAN*	TRUE / FALSE *SALAH / BETUL*
1	A compiler is a utility program that converts source code programs from assembly language into machine language. *Sebuah penterjemah adalah program utiliti yang menukarkan program kod sumber dari bahasa himpunan kepada bahasa mesin.*	
2	The sign flag is set when the result of an arithmetic or logical operation generates a negative result. *Bendera tanda ditetapkan apabila hasil aritmetik atau operasi logik menghasilkan hasil negatif.*	
3	EPROM is permanently burned into a chip and cannot be erased. *EPROM adalah terbakar secara kekal ke dalam cip dan tidak boleh dipadam.*	
4	PCI bus provides a connecting bridge between the CPU and other systems devices such as hard drives, memory, video controllers, sound cards, and network controllers. *PCI bus menyediakan jambatan penyambungan antara CPU dan peranti sistem lain seperti cakera keras, memori, pengawal video, kad bunyi, dan pengawal rangkaian.*	
5	EBP register is rarely used for ordinary arithmetic or data transfer. *Pendaftar EBP jarang digunakan untuk aritmetik biasa atau pemindahan data.*	

6	Registers are high-speed storage locations directly inside the CPU, designed to be accessed at much higher than conventional memory. *Pendaftar adalah lokasi penyimpanan berkelajuan tinggi yang terletak secara langsung di dalam CPU, direka untuk diakses pada kadar yang jauh lebih tinggi daripada memori konvensional.*	
7	Address bus uses binary signals to synchronize actions of all devices attached to the system bus. *Bas alamat menggunakan isyarat binari untuk menyelaraskan tindakan semua peranti yang dilampirkan kepada bas sistem.*	
8	Control Unit coordinates the sequencing of steps involved in executing machine instructions. *Unit Kawalan menyelaraskan urutan langkah-langkah yang terlibat dalam menjalankan arahan mesin.*	
9	Assembly language is portable and can be compiled and run on a wide variety of computer systems. *Bahasa perhimpunan adalah mudah alih dan boleh dijalankan pada pelbagai sistem komputer setelah dikompilasi.*	
10	Two popular Linux-based assemblers are GAS and NASM. *Dua pengkompil asas Linux yang popular adalah GAS dan NASM.*	

SECTION B / *BAHAGIAN B*

65 MARKS / *65 MARKAH*

SUBJECTIVE QUESTIONS / *SOALAN SUBJEKTIF*

Answer all questions. / *Jawab semua soalan.*

1. What is equivalent of -37_{10} in 8-bit two's complement representation? Show your calculation.
Apakah yang setara dengan -37_{10} dalam perwakilan pelengkap dua 8-bit? Tunjukkan pengiraan anda. **[4M]**

[3M]

Final Answer / *Jawapan Akhir* = _____ [binary] **[1M]**

2. What is equivalent of $81AC2_{16}$ in two's complement hexadecimal? Show your calculation.
Apakah yang setara dengan $81AC2_{16}$ dalam heksadesimal pelengkap dua? Tunjukkan pengiraan anda. **[4M]**

[3M]

Final Answer / *Jawapan Akhir* = _____ [hexadecimal] **[1M]**

3. Perform the following addition and show how the bits (CF, ZF, AF, PF, OF, and SF) of the flag register is affected by the following operations.
Lakukan penambahan berikut dan tunjukkan bagaimana bit (CF, ZF, AF, PF, OF, dan SF) dari daftar bendera dipengaruhi oleh operasi berikut. **[6M]**

3211h + 7953h **[3M]**	CF = __	**[0.5M]**
	PF = __	**[0.5M]**
	AF = __	**[0.5M]**
	ZF = __	**[0.5M]**
	SF = __	**[0.5M]**
	OF = __	**[0.5M]**

4. What is equivalent of signed decimal -100_{10} to binary representation? Show your working.
Apakah yang setara dengan nombor decimal bertanda -100_{10} *kepada perwakilan binari? Tunjukkan kerja anda.* **[4M]**

[3M]	
Final Answer / *Jawapan Akhir* = _____ [binary]	**[1M]**

5. Given two decimal numbers, X and Y. Suppose **X = 243d** and **Y = 132d**. Convert each of them into 8-bit binary numbers and then show how does a computer performs **X - Y** operation. Show how the flags register is affected after the operation.
*Diberi dua nombor perpuluhan, X dan Y. Andaikan **X = 243d** dan **Y = 132d**. Tukarkan masing-masing menjadi nombor perduaan 8-bit dan kemudian tunjukkan bagaimana komputer melakukan operasi **X - Y**. Tunjukkan bagaimana daftar bendera terjejas selepas operasi.* **[6M]**

X – Y		
[3M]	CF = __	**[0.5M]**
	PF = __	**[0.5M]**
	AF = __	**[0.5M]**
	ZF = __	**[0.5M]**
	SF = __	**[0.5M]**
	OF = __	**[0.5M]**

6. What is the clock cycle time in **microseconds** of an **80 KHz** processor? Show your working.
*Apakah masa kitaran jam dalam **mikrosaat** bagi pemproses **80 KHz**? Tunjukkan kerja anda.* **[2M]**
Note: Clock rate table is attached in APPENDIX B.
Nota: Jadual kadar jam dilampirkan pada APPENDIX B.

[2M]

7. Indicate the content of register BL and CL (in hex value) and status flag of CF after the execution of the following program fragment.
Nyatakan kandungan daftar BL dan CL (dalam nilai hex) dan bendera status CF setelah pelaksanaan keratan program berikut. **[6M]**

```
mov     cx, 0F1h
sar     cl, 2
mov     bl, 52h
shr     bl, 1
shl     bl, 4
```

Notes: Can use this boxes for calculation purposes. Marks will be given only for the final answer.
Nota: Boleh menggunakan petak ini untuk tujuan pengiraan. Markah akan diberikan hanya untuk jawapan akhir.

mov cx, 0F1h **CX** ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

CL ☐☐☐☐☐☐☐☐ ☐ **CF**

sar cl, 2 **CL** ☐☐☐☐☐☐☐☐ ☐ **CF**

mov bl, 52h **BL** ☐☐☐☐☐☐☐☐ ☐ **CF**

shr bl, 1 **BL** ☐☐☐☐☐☐☐☐ ☐ **CF**

CF ☐ ☐☐☐☐☐☐☐☐ **BL**

shl bl, 4 **CF** ☐ ☐☐☐☐☐☐☐☐ **BL**

BL = _____ [hexadecimal] **[2M]** CL = _____ [hexadecimal] **[2M]** CF = ___ **[2M]**

8. Refer to the following data declaration:
Rujuk kepada pengisytiharaan data berikut:

```
.data
Lapras          BYTE        2d, 1011b, 0A1h, 40d
Gyarados        WORD        7ACCh, 0F111h, 101100111010b
Vaporeon        SWORD       -266d, -1175
Bagon           DWORD       111111d, 222222d, 333333d, 444444d
```

For each of the following statements, state whether the instruction is **VALID** or **NOT VALID**.
*Bagi setiap pernyataan berikut, nyatakan sama ada arahan itu **SAH** atau **TIDAK SAH**.* **[5M]**

Note: x86 instruction set is attached in APPENDIX C.
Nota: Set arahan x86 dilampirkan pada APPENDIX C.

i) mov dx, Lapras = _____ **[0.5M]**

ii) mov bx, Gyarados = _____ **[0.5M]**

iii) movzx ecx, Bagon = _____ **[0.5M]**

iv) movsx eax, Vaporeon = _____ **[0.5M]**

v) movzx Gyarados, cl = _____ **[0.5M]**

vi) cmp Lapras, Gyarados = _____ **[0.5M]**

vii) add Lapras, 123d = _____ **[0.5M]**

viii) sub esi, 10010101011b = _____ **[0.5M]**

ix) mov cs, bx = _____ **[0.5M]**

x) mov ss, 75d = _____ **[0.5M]**

9. Show the value of the flag bits after the following instructions are executed:
Tunjukkan nilai bit bendera selepas arahan berikut dilaksanakan: **[2M]**

```
.code
mov     ecx, 125h
mov     edx, 25h
cmp     ecx, edx
```

ZF = _____ **[1M]** CF = _____ **[1M]**

10. Show the value of the flag bits after the following instructions are executed:
Tunjukkan nilai bit bendera selepas arahan berikut dilaksanakan: **[3M]**

```
mov     cl, 11101010b
mov     dl, 01010010b
test    cl, 01001011b       ;  .............[1]
test    dl, 01011011b       ;  .............[2]
```

[1]	[2]
ZF = _____ **[1.5M]**	ZF = _____ **[1.5M]**

11. Refer to the following fragments:
Rujuk keratan aturcara berikut:

```
.data
     Var1    BYTE     100d, 99d, 98, 97d, 96d
     Var2    WORD     1234h, 5678h, 9ABCh, DEF0h, 22AAh
     Var3    DWORD    0AAAAAAh, 0CCCCCCh, 0EEEEEEh
.code
main PROC
     mov   edi, 2
     mov   bl,  Var1[edi]          ;     .............[1]
     mov   cx,  Var2[edi * 2]      ;     .............[2]
     mov   edx, Var3[edi * 4]      ;     .............[3]
```

Suppose the address of the data segment starts at address **0AA20100h.** What is the effective address of the source operand for instruction labeled **[1]**, **[2]**, and **[3]**? Show your calculation.
*Andaikan alamat segmen data bermula pada alamat **0AA20100h**. Apakah alamat efektif bagi kendalian sumber bagi arahan **[1]**, **[2]**, dan **[3]**? Tunjukkan pengiraan anda.* **[3M]**

[1] Effective Address / *Alamat Efektif* = _____ **[1M]**

[2] Effective Address / *Alamat Efektif* = _____ **[1M]**

[3] Effective Address / *Alamat Efektif* = _____ **[1M]**

12. Show the content of the individual bytes allocated in memory (in hexadecimal) for the following data declarations. Assume a computer with 32-bit address bus, and that the physical address of **Kitten** is **0A211080h**.

*Tunjukkan kandungan setiap bait yang diperuntukkan dalam ingatan (dalam heksadesimal) untuk pengisytiharan data berikut. Andaikan komputer yang mempunyai 32-bit bas alamat, dan alamat fizikal **'Kitten'** adalah **0A211080h**.* **[20M]**

Note: ASCII Table attached in APPENDIX A.

Nota: Jadual ASCII dilampirkan pada APPENDIX A.

```
.data
Kitten          WORD        7196h, 246h, 0EEEh
Align 8
Lion            SDWORD      -1201d, -111d, +953
Chelsea         LABEL       DWORD
RealMadrid      LABEL       WORD
Align 4
Duck            BYTE        2 DUP ("A~"), 2 DUP (98d)
Cow             BYTE        0C1h, 75h, 2 DUP (6 DUP (1 DUP (?)))
Ant             EQU         <"I Win Now", 0>
Bear            TEXTEQU     <"Mcdonalds!">
Align 2
Eagle           DWORD       Duck
Mermaid         BYTE        Ant
Shark           EQU         4*2
Sum = 10h
```

Memory Layout / *Susun Atur Memori*:

LABEL	OFFSET	CONTENT	MEMORY (HEX)
Kitten	0A211080	7196h	96
	0A211081		71
	0A211082	246h	46
	0A211083		02

	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]

	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]
	0A2110____			[0.2M]

What will be the content of register (in hex) if the following instruction is executed?
Apakah kandungan daftar (dalam heksa) jika arahan berikut dilaksanakan.

i)	mov	ebx, TYPE Lion	;	EBX = _____	**[0.5M]**
ii)	mov	ebx, TYPE Kitten	;	EBX = _____	**[0.5M]**
iii)	mov	ebx, LENGTHOF Cow	;	EBX = _____	**[0.5M]**
iv)	mov	ebx, LENGTHOF Duck	;	EBX = _____	**[0.5M]**
v)	mov	ebx, SIZEOF Duck	;	EBX = _____	**[0.5M]**
vi)	mov	ebx, SIZEOF Cow	;	EBX = _____	**[0.5M]**
vii)	mov	ebx, OFFSET Kitten	;	EBX = _____	**[0.5M]**
viii)	mov	ebx, OFFSET Lion	;	EBX = _____	**[0.5M]**
ix)	mov	ebx, OFFSET Duck	;	EBX = _____	**[0.5M]**
x)	mov	ebx, OFFSET Cow	;	EBX = _____	**[0.5M]**
xi)	mov	ebx, OFFSET Eagle - 2	;	EBX = _____	**[0.5M]**
xii)	mov	ebx, OFFSET Mermaid + 3	;	EBX = _____	**[0.5M]**
xiii)	mov	ebx, Chelsea	;	EBX = _____	**[0.5M]**
xiv)	mov	cx, RealMadrid	;	CX = _____	**[0.5M]**
xv)	mov	cl, BYTE PTR Lion	;	CL = _____	**[0.5M]**
xvi)	mov	eax, DWORD PTR Cow	;	EAX = _____	**[0.5M]**
xvii)	mov	ax, WORD PTR Eagle	;	AX = _____	**[0.5M]**
xviii)	mov	ecx, [Lion + 3]	;	ECX = _____	**[0.5M]**
xix)	mov	ah, [Cow - 4]	;	AH = _____	**[0.5M]**
xx)	mov	eax, Sum	;	EAX = _____	**[0.5M]**

SECTION C / *BAHAGIAN C*

25 MARKS / *25 MARKAH*

PROGRAMMING / *PENGATURCARAAN*

Answer all questions.

Jawab semua soalan.

Programming 1 / *Pengaturcaraan 1*

Complete a program to solve the arithmetic expression below:

Lengkapkan satu atur cara untuk menyelesaikan ungkapan aritmetik di bawah:

```
Sum = -Val1 - Val2 + 1 + (Val3 / (Val4 - Val5)) * 3
```

You should use the following information for your code:

Anda harus menggunakan maklumat berikut untuk kod anda:

- Declare **Val1** with 32-bit unsigned integer variable and initialize it with 70 decimal value.
  *Isytiharkan **Val1** dengan pembolehubah integer tidak bertanda 32-bit dan mulakannya dengan nilai perpuluhan 70.*
- Declare **Val2** with 32-bit unsigned integer variable and initialize it with 20 decimal value.
  *Isytiharkan **Val2** dengan pembolehubah integer tidak bertanda 32-bit dan mulakannya dengan nilai perpuluhan 20.*
- Declare **Val3** with 32-bit unsigned integer variable and initialize it with 60 decimal value.
  *Isytiharkan **Val3** dengan pembolehubah integer tidak bertanda 32-bit dan mulakannya dengan nilai perpuluhan 60.*
- Declare **Val4** with 32-bit unsigned integer variable and initialize it with 6 decimal value.
  *Isytiharkan **Val4** dengan pembolehubah integer tidak bertanda 32-bit dan mulakannya dengan nilai perpuluhan 6.*
- Declare **Val5** with 32-bit unsigned integer variable and initialize it with 4 decimal value.
  *Isytiharkan **Val5** dengan pembolehubah integer tidak bertanda 32-bit dan mulakannya dengan nilai perpuluhan 4.*

[10M]

Note: x86 Instruction Set attached in APPENDIX C.

Nota: Set Arahan x86 dilampirkan pada APPENDIX C.

```
TITLE MASM Template (Template.asm)
INCLUDE Irvine32.inc

;       Sum = -Val1 - Val2 + 1 + (Val3 / (Val4 - Val5)) * 3     |       Question
;1-     Sum = -Val1 - Val2 + 1 + (Val3 / (EBX)) * 3             |       EBX = (Val4 - Val5)
;2-     Sum = -Val1 - Val2 + 1 + (EAX) * 3                      |       EAX = (Val3 / EBX)
;3-     Sum = -Val1 - Val2 + 1 + EAX                            |       EAX = (EAX * 3)
;4-     Sum = (Val1) - Val2 + 1 + EAX                           |       Val1 = -Val1
;5-     Sum = (ECX) + 1 + EAX                                   |       ECX = Val1 - Val2
;6-     Sum = (ECX) + EAX                                       |       ECX = ECX + 1
;7-     Sum = (ECX)                                             |       ECX = ECX + EAX

.data
Val1            DWORD           _____ [0.5M]
Val2            DWORD           _____ [0.5M]
Val3            DWORD           _____ [0.5M]
Val4            DWORD           _____ [0.5M]
Val5            DWORD           _____ [0.5M]
Sum             DWORD           ?

.code
main PROC
;1- ANSWER IN EBX
        mov     ebx, _____ [0.5M]
        sub     ebx, _____ [0.5M]

;2- ANSWER IN EAX
        mov     edx, 0d                         ; EDX = Clear the register
        mov     eax, _____ [0.5M]
        div      _____ [0.5M]           ; EAX = EAX / EBX

;3- ANSWER IN EAX
        mov     ecx, _____ [0.5M]
        mul      _____ [0.5M]           ; EAX = EAX * 3d
```

```
;4- ANSWER IN VAL1
        neg     _____________________ [0.5M]      ; VAL1 = -VAL1

;5- ANSWER IN ECX
        mov     ecx, _____________________ [0.5M]
        sub     ecx, _____________________ [0.5M]

;6- ANSWER IN ECX
        inc     _____________________ [0.5M]

;7- ANSWER IN ECX
        add     ecx, _____________________ [0.5M]

;SAVE THE ANSWER IN SUM
        mov     sum, _____________________ [0.5M]

;DISPLAY ANSWER IN SIGNED DECIMAL
        mov     eax, _____________________ [0.5M]
        call    _____________________ [0.5M]
        call    Crlf

  exit
main ENDP
END main
```

What is the output of this program?
Apakah output program ini?

```
                                                              [0.5M]
```

Programming 2 / *Pengaturcaraan 2*

Implement the following C++ Programming in assembly language. Your program **MUST** use **LOOP**, **JE** and **JB** instructions. Your program should follow exactly as the sample output, do use new line instructions in assembly language to replace **endl** that have been used in the following C++ Programming.

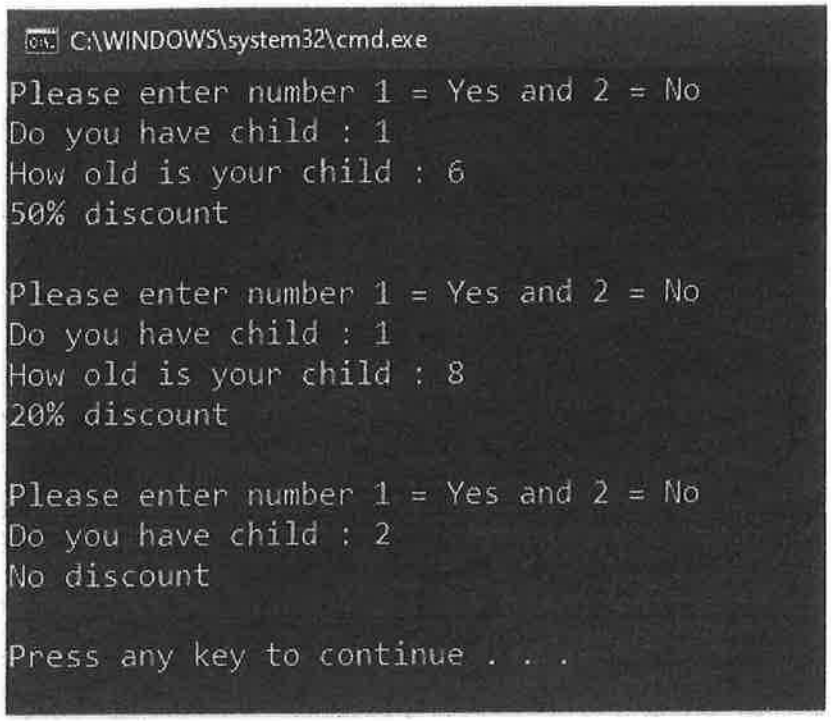
*Laksanakan Pengaturcaraan C++ berikut dalam bahasa perhimpunan. Program anda **MESTI** menggunakan arahan **LOOP**, **JE** dan **JB**. Program anda harus mengikuti betul-betul seperti contoh output, gunakan arahan baris baru dalam bahasa himpunan untuk menggantikan **endl** yang telah digunakan dalam Pengaturcaraan C++ berikut.* **[15M]**

Note: Conditional Jumps Table attached in APPENDIX D.
Nota: Jadual Lompat Bersyarat dilampirkan pada APPENDIX D.

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5
6      int child, age;
7
8      for(int i = 0; i < 3; i++){
9
10         cout << "Please enter number 1 = Yes and 2 = No";
11         cout << endl;
12
13         cout << "Do you have child : ";      //User enter 1 for yes and 2 for no
14         cin >> child;
15
16         if(child == 1){
17
18             cout << "How old is your child : "; //User enter age of their child
19             cin >> age;
20
21             if(age < 8){
22                 cout << "50% discount";
23                 cout << endl;
24             }else{
25                 cout << "20% discount";
26                 cout << endl;
27             }
28
29         }else{
30             cout << "No Discount";
31             cout << endl;
32         }
33         cout << endl;
34     }
35 }
```

Sample output should be same as below:

Output sampel hendaklah sama seperti di bawah:




```
TITLE MASM Template (Template.asm)
INCLUDE Irvine32.inc
```

.data **[3M]**

child	_____	?	[0.25M]
age	_____	_____	[0.5M]
prompt1	_____	"Please enter number 1 = Yes and 2 = No", 0	[0.25M]
prompt2	byte	_____	[0.25M]
prompt3	_____	_____	[0.5M]
msg1	_____	"50% discount", 0	[0.25M]
msg2	_____	_____	[0.5M]
msg3	_____	_____	[0.5M]

```
.code                                                    [12M]
main PROC
```

	_____ ; set loop	[0.5M]
start:	_____ ; prompt1	[0.5M]
	_____	[0.5M]
	call crlf	
	_____ ; prompt2	[0.5M]
	_____	[0.5M]
	_____ ; get input	[0.5M]
	_____ ; compare	[0.5M]
	_____ ; if has child	[0.5M]
	_____ ; if no child display msg3	[0.5M]
	_____	[0.5M]
	call crlf	
	_____ ; go to last	[0.5M]
haschild:	_____ ; prompt3	[0.5M]
	_____	[0.5M]
	_____ ; get input	[0.5M]
	_____ ; compare	[0.5M]
	_____ ; if age < 8	[0.5M]

```
                _____ ; if age >= 8 display msg2
                _____                                   [0.5M]
                call crlf                               [0.5M]
                _____ ; go to last
                                                        [0.5M]
dis50:          _____ ; display msg1
                _____                                   [0.5M]
                call crlf                               [0.5M]
                _____ ;  go to last
                                                        [0.5M]
last:           _____
                _____ ;  repeat to start                [0.5M]
                                                        [0.5M]
```

```
  exit
main ENDP
END main
```

END OF QUESTIONS / *SOALAN TAMAT*

APPENDIX A

ASCII TABLE / *JADUAL ASCII*

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

APPENDIX B

CLOCK RATE TABLE / *JADUAL KADAR JAM*

Clock Rate / Frequency (SI Units)	Cycle Time	Name	Factor	Number of Cycles / Waves per second
1 hertz (Hz)	1 cycle / sec	second	1	1
1 kilohertz (KHz)	10^3 cycles / sec	millisecond	10^{-3}	1 / 1000
1 megahertz (MHz)	10^6 cycles / sec	microsecond	10^{-6}	1 / 1000000
1 gigahertz (GHz)	10^9 cycles / sec	nanosecond	10^{-9}	1 / 1000000000
1 terahertz (THz)	10^{12} cycles / sec	picosecond	10^{-12}	1 / 1000000000000
1 petahertz (PHz)	10^{15} cycles / sec	femtosecond	10^{-15}	1 / 1000000000000000
1 exahertz (EHz)	10^{18} cycles / sec	attosecond	10^{-18}	1 / 1000000000000000000

APPENDIX C

x86 INSTRUCTION SET / *SET ARAHAN x86*

ADD — **Add**

O	D	I	S	Z	A	P	C
*			*	*	*	*	*

A source operand is added to a destination operand, and the sum is stored in the destination. Operands must be the same size.

Instruction formats:

```
ADD   reg,reg          ADD   reg,imm
ADD   mem,reg          ADD   mem,imm
ADD   reg,mem          ADD   accum,imm
```

CMP — **Compare**

O	D	I	S	Z	A	P	C
*			*	*	*	*	*

Compares the destination to the source by performing an implied subtraction of the source from the destination.

Instruction formats:

```
CMP   reg,reg          CMP   reg,imm
CMP   mem,reg          CMP   mem,imm
CMP   reg,mem          CMP   accum,imm
```

DEC

Decrement

O	D	I	S	Z	A	P	C
*			*	*	*	*	

Subtracts 1 from an operand. Does not affect the Carry flag.

Instruction formats:

```
DEC   reg                    DEC   mem
```

DIV

Unsigned Integer Divide

O	D	I	S	Z	A	P	C
?			?	?	?	?	?

Performs either 8-, 16-, or 32-bit unsigned integer division. If the divisor is 8 bits, the dividend is AX, the quotient is AL, and the remainder is AH. If the divisor is 16 bits, the dividend is DX:AX, the quotient is AX, and the remainder is DX. If the divisor is 32 bits, the dividend is EDX:EAX, the quotient is EAX, and the remainder is EDX.

Instruction formats:

```
DIV   reg                    DIV   mem
```

IMUL

Signed Integer Multiply

O	D	I	S	Z	A	P	C
*			?	?	?	?	*

Performs a signed integer multiplication on AL, AX, or EAX. If the multiplier is 8 bits, the multiplicand is AL and the product is AX. If the multiplier is 16 bits, the multiplicand is AX and the product is DX:AX. If the multiplier is 32 bits, the multiplicand is EAX and the product is EDX:EAX. The Carry and Overflow flags are set if a 16-bit product extends into AH, or a 32-bit product extends into DX, or a 64-bit product extends into EDX.

Instruction formats:

Single operand:

```
IMUL  r/m8                   IMUL  r/m16
IMUL  r/m32
```

Two operands:

```
IMUL  r16,r/m16              IMUL  r16,imm8
IMUL  r32,r/m32              IMUL  r32,imm8
IMUL  r16,imm16              IMUL  r32,imm32
```

Three operands:

```
IMUL  r16,r/m16,imm8         IMUL  r16,r/m16,imm16
IMUL  r32,r/m32,imm8         IMUL  r32,r/m32,imm32
```

INC

Increment

O	D	I	S	Z	A	P	C
*			*	*	*	*	

Adds 1 to a register or memory operand.

Instruction formats:

```
INC   reg                    INC   mem
```

LOOP

Loop

O	D	I	S	Z	A	P	C

Decrements ECX and jumps to a short label if ECX is not equal to zero. The destination must be −128 to +127 bytes from the current location.

Instruction formats:

```
LOOP  shortlabel             LOOPW shortlabel
```

MOV

Move

O	D	I	S	Z	A	P	C

Copies a byte or word from a source operand to a destination operand.

Instruction formats:

```
MOV  reg,reg                 MOV  reg,imm
MOV  mem,reg                 MOV  mem,imm
MOV  reg,mem                 MOV  mem16,segreg
MOV  reg16,segreg            MOV  segreg,mem16
MOV  segreg,reg16
```

MOVSX

Move with Sign-Extend

O	D	I	S	Z	A	P	C

Copies a byte or word from a source operand to a destination register and sign-extends into the upper bits of the destination. This instruction is used to copy an 8-bit or 16-bit operand into a larger destination.

Instruction formats:

```
                             MOVSX reg32,reg8
MOVSX reg32,reg16            MOVSX reg32,mem16
MOVSX reg16,reg8             MOVSX reg16,m8
```

MOVZX — Move with Zero-Extend

O	D	I	S	Z	A	P	C

Copies a byte or word from a source operand to a destination register and zero-extends into the upper bits of the destination. This instruction is used to copy an 8-bit or 16-bit operand into a larger destination.

Instruction formats:

```
MOVZX  reg32,reg16          MOVZX  reg32,reg8
MOVZX  reg16,reg8           MOVZX  reg32,mem16
                            MOVZX  reg16,m8
```

MUL — Unsigned Integer Multiply

O	D	I	S	Z	A	P	C
*			?	?	?	?	*

Multiplies AL, AX, or EAX by a source operand. If the source is 8 bits, it is multiplied by AL and the product is stored in AX. If the source is 16 bits, it is multiplied by AX and the product is stored in DX:AX. If the source is 32 bits, it is multiplied by EAX and the product is stored in EDX:EAX.

Instruction formats:

```
MUL reg                     MUL mem
```

NEG — Negate

O	D	I	S	Z	A	P	C
*			*	*	*	*	*

Calculates the twos complement of the destination operand and stores the result in the destination.

Instruction formats:

```
NEG reg                     NEG mem
```

SUB — Subtract

O	D	I	S	Z	A	P	C
*			*	*	*	*	*

Subtracts the source operand from the destination operand.

Instruction formats:

```
SUB  reg,reg                SUB  reg,imm
SUB  mem,reg                SUB  mem,imm
SUB  reg,mem                SUB  accum,imm
```

APPENDIX D

CONDITIONAL JUMPS TABLE / *JADUAL LOMPAT BERSYARAT*

TABLE D1: JUMPS BASED ON SPECIFIC FLAG VALUES

Mnemonic	Description	Flags / Registers
JZ	Jump if zero	ZF = 1
JNZ	Jump if not zero	ZF = 0
JC	Jump if carry	CF = 1
JNC	Jump if not carry	CF = 0
JO	Jump if overflow	OF = 1
JNO	Jump if not overflow	OF = 0
JS	Jump if signed	SF = 1
JNS	Jump if not signed	SF = 0
JP	Jump if parity (even)	PF = 1
JNP	Jump if not parity (odd)	PF = 0

TABLE D2: JUMPS BASED ON EQUALITY

Mnemonic	Description
JE	Jump if equal (*leftOp* = *rightOp*)
JNE	Jump if not equal (*leftOp* ≠ *rightOp*)
JCXZ	Jump if CX = 0
JECXZ	Jump if ECX = 0
JRCXZ	Jump if RCX = 0 (64-bit mode)

TABLE D3: JUMPS BASED ON UNSIGNED COMPARISON

Mnemonic	Description
JA	Jump if above (if *leftOp* > *rightOp*)
JNBE	Jump if not below or equal (same as JA)
JAE	Jump if above or equal (if *leftOp* ≥ *rightOp*)
JNB	Jump if not below (same as JAE)
JB	Jump if below (if *leftOp* < *rightOp*)
JNAE	Jump if not above or equal (same as JB)
JBE	Jump if below or equal (if *leftOp* ≤ *rightOp*)
JNA	Jump if not above (same as JBE)

TABLE D4: JUMPS BASED ON SIGNED COMPARISON

Mnemonic	Description
JG	Jump if greater (if *leftOp* > *rightOp*)
JNLE	Jump if not less than or equal (same as JG)
JGE	Jump if greater than or equal (if *leftOp* ≥ *rightOp*)
JNL	Jump if not less (same as JGE)
JL	Jump if less (if *leftOp* < *rightOp*)
JNGE	Jump if not greater than or equal (same as JL)
JLE	Jump if less than or equal (if *leftOp* ≤ *rightOp*)
JNG	Jump if not greater (same as JLE)