

Contrôle 2

Architecture des ordinateurs

Durée : 1 h 30

Exercice 1 (4 points)

Codez les instructions suivantes en langage machine 68000, vous détaillerez les différents champs puis vous exprimerez le résultat final sous forme hexadécimale en précisant la taille des mots supplémentaires lorsque le cas se présente.

1. MOVE.B -(A5), (A4)
2. ADDA.L -1(A3), A2
3. MOVE.W #34, 34
4. MOVE.L #\$51, 26(A3, A1.W)

Exercice 2 (4 points)

Vous indiquerez après chaque instruction, le nouveau contenu des registres (sauf le PC) et/ou de la mémoire qui viennent d'être modifiés. Vous utiliserez la représentation hexadécimale.

Attention : La mémoire et les registres sont réinitialisés à chaque nouvelle instruction.

Valeurs initiales : D0 = \$FFFFFFFE A0 = \$00005000 PC = \$00006000

 D1 = \$FFFF0005 A1 = \$00005008

 D2 = \$FFFFFF00 A2 = \$00005010

 \$005000 54 AF 18 B9 E7 21 48 C0

 \$005008 C9 10 11 C8 D4 36 1F 88

 \$005010 13 79 01 80 42 1A 2D 48

1. MOVE.W #\$27, -(A1)
2. MOVE.L D2, 4(A2, D0.L)
3. MOVE.B \$6006(PC, D2.L), \$5010
4. MOVE.W -1(A2, D1.W), 2(A0)

Exercice 3 (3 points)

Donnez le résultat des additions hexadécimales suivantes, ainsi que le contenu des bits N, Z, V et C du registre d'état.

1. \$3D + \$E9 opération en .B
2. \$6AB4 + \$3FC6 opération en .W

Exercice 4 (2 points)

Réalisez le sous-programme **Add128** qui réalise une addition sur 128 bits en quelques lignes seulement (pas plus de cinq lignes).

Entrées : **D3:D2:D1:D0** = Entier sur 128 bits (**D0** étant les 32 bits de poids faible).

D7:D6:D5:D4 = Entier sur 128 bits (**D4** étant les 32 bits de poids faible).

Sorties : **D3:D2:D1:D0** = **D3:D2:D1:D0** + **D7:D6:D5:D4**

Exercice 5 (3 points)

Réalisez le sous-programme **GetValue** en fonction des entrées-sorties ci-dessous (hormis le registre de sortie, aucun registre ne sera modifié en sortie du sous-programme) :

Entrée : **D1.W** = Entier signé sur 16 bits.

Sorties : **D0.L** = 1 si **D1.W** est négatif.

D0.L = 2 si **D1.W** est nul.

D0.L = 3 dans tous les autres cas.

Exercice 6 (4 points)

Soit les deux instructions suivantes :

- `MOVEM.L D2/D1/A1/A5, -(A7)`
- `MOVEM.L (A7)+, A5/A1/D1/D2`

1. Laquelle des deux permet d'empiler les registres ?
2. Dans quel ordre seront-ils empilés ?
3. Dans quel ordre seront-ils dépilés ?
4. Choisissez la proposition exacte :
Après l'exécution d'une instruction RTS, le pointeur de pile est :
 - incrémenté de deux ;
 - décrémenté de deux ;
 - incrémenté de quatre ;
 - décrémenté de quatre ;
 - inchangé.
5. Si un programmeur commet l'erreur d'utiliser une instruction JMP à la place d'une instruction JSR, quel problème cela peut-il poser ?

Integer Instructions

MOVE

Move Data from Source to Destination
(M68000 Family)

MOVE

Operation: Source → Destination

Assembler

Syntax: MOVE < ea > , < ea >

Attributes: Size = (Byte, Word, Long)

Description: Moves the data at the source to the destination location and sets the condition codes according to the data. The size of the operation may be specified as byte, word, or long. Condition Codes:

X	N	Z	V	C
—	*	*	0	0

X — Not affected.

N — Set if the result is negative; cleared otherwise.

Z — Set if the result is zero; cleared otherwise.

V — Always cleared.

C — Always cleared.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		0		SIZE		DESTINATION				SOURCE					
				REGISTER		MODE				MODE				REGISTER	

Instruction Fields:

Size field—Specifies the size of the operand to be moved.

01 — Byte operation

11 — Word operation

10 — Long operation

MOVE**Move Data from Source to Destination
(M68000 Family)****MOVE**

Destination Effective Address field—Specifies the destination location. Only data alterable addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An	—	—	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	—	—
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d ₁₆ ,An)	101	reg. number:An	(d ₁₆ ,PC)	—	—
(d ₈ ,An,Xn)	110	reg. number:An	(d ₈ ,PC,Xn)	—	—

MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An	(bd,PC,Xn)*	—	—
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	—	—
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	—	—

*Can be used with CPU32.

Source Effective Address field—Specifies the source operand. All addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An	001	reg. number:An	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	111	100
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d ₁₆ ,An)	101	reg. number:An	(d ₁₆ ,PC)	111	010
(d ₈ ,An,Xn)	110	reg. number:An	(d ₈ ,PC,Xn)	111	011

MC68020, MC68030, and MC68040 only

(bd,An,Xn)**	110	reg. number:An	(bd,PC,Xn)**	111	011
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	111	011
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	111	011

*For byte size operation, address register direct is not allowed.

**Can be used with CPU32.

NOTE

Most assemblers use MOVEA when the destination is an address register.

MOVEQ can be used to move an immediate 8-bit value to a data register.

ADDA

Add Address (M68000 Family)

ADDA

Operation: Source + Destination → Destination

Assembler

Syntax: ADDA < ea > , An

Attributes: Size = (Word, Long)

Description: Adds the source operand to the destination address register and stores the result in the address register. The size of the operation may be specified as word or long. The entire destination address register is used regardless of the operation size.

Condition Codes:

Not affected.

Instruction Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	REGISTER			OPMODE			EFFECTIVE ADDRESS					
										MODE					REGISTER

Instruction Fields:

Register field—Specifies any of the eight address registers. This is always the destination.

Opmode field—Specifies the size of the operation.

011— Word operation; the source operand is sign-extended to a long operand and the operation is performed on the address register using all 32 bits.

111— Long operation.

Effective Address field—Specifies the source operand. All addressing modes can be used as listed in the following tables:

Addressing Mode	Mode	Register	Addressing Mode	Mode	Register
Dn	000	reg. number:Dn	(xxx).W	111	000
An	001	reg. number:An	(xxx).L	111	001
(An)	010	reg. number:An	#<data>	111	100
(An) +	011	reg. number:An			
– (An)	100	reg. number:An			
(d ₁₅ ,An)	101	reg. number:An	(d ₁₆ ,PC)	111	010
(d ₈ ,An,Xn)	110	reg. number:An	(d ₈ ,PC,Xn)	111	011

MC68020, MC68030, and MC68040 only

(bd,An,Xn)*	110	reg. number:An	(bd,PC,Xn)*	111	011
([bd,An,Xn],od)	110	reg. number:An	([bd,PC,Xn],od)	111	011
([bd,An],Xn,od)	110	reg. number:An	([bd,PC],Xn,od)	111	011

*Can be used with CPU32

M68000 FAMILY PROGRAMMER'S REFERENCE MANUAL

MOTOROLA

BRIEF EXTENSION WORD FORMAT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/A	REGISTER				W/L	0	0	0	DISPLACEMENT INTEGER						

(a) MC68000, MC68008, and MC68010

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/A	REGISTER				W/L	SCALE	0	DISPLACEMENT INTEGER							

(b) CPU32, MC68020, MC68030, and MC68040

Table 2-1. Instruction Word Format Field Definitions

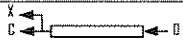
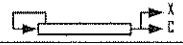
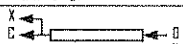


Field	Definition
Instruction	
Mode	Addressing Mode
Register	General Register Number
Extensions	
D/A	Index Register Type 0 = Dn 1 = An
W/L	Word/Long-Word Index Size 0 = Sign-Extended Word 1 = Long Word
Scale	Scale Factor 00 = 1 01 = 2 10 = 4 11 = 8
BS	Base Register Suppress 0 = Base Register Added 1 = Base Register Suppressed
IS	Index Suppress 0 = Evaluate and Add Index Operand 1 = Suppress Index Operand
BD SIZE	Base Displacement Size 00 = Reserved 01 = Null Displacement 10 = Word Displacement 11 = Long Displacement
I/IS	Index/Indirect Selection Indirect and Indexing Operand Determined in Conjunction with Bit 6, Index Suppress

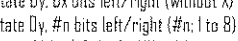
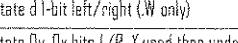
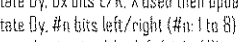
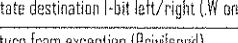
For effective addresses that use a full extension word format, the index suppress (IS) bit and the index/indirect selection (I/IS) field determine the type of indexing and indirect action. Table 2-2 lists the index and indirect operations corresponding to all combinations of IS and I/IS values.

EASy68K Quick Reference v2.1

www.easy68k.com

Copyright © 2004-2009 By: Chuck Kelly

Opcode	Size	Operand	CCR	Effective Address s=source, d=destination, e=either, i=displacement													Operation	Description
		s,d	XN2VC	Dn	An	(An)	(An)+	-(An)	(i,An)	(i,An,Rn)	abs.W	abs.L	(i,PC)	(i,PC,Rn)	#n			
ABCD	B	Dy,Dx -(Ay),-(Ax)	*U*U*	e	-	-	-	-	-	-	-	-	-	-	-	-	$Dy_{10} + Dx_{10} + X \rightarrow Dx_{10}$ $-(Ay)_{10} + -(Ax)_{10} + X \rightarrow -(Ax)_{10}$	BCD destination + BCD source + eXtend Z cleared if result not 0 unchanged otherwise
ADD ⁴	BWL	s,Dn Dn,d	*****	e	s	s	s	s	s	s	s	s	s	s	s	s ⁴	$s + Dn \rightarrow Dn$ $Dn + d \rightarrow d$	Add binary (ADD or ADDQ is used when source is #n. Prevent ADDQ with #n.L)
ADDA ⁴	WL	s,An	-----	s	e	s	s	s	s	s	s	s	s	s	s	s	$s + An \rightarrow An$	Add address (W sign-extended to .L)
ADDI ⁴	BWL	#n,d	*****	d	-	d	d	d	d	d	d	d	-	-	-	s	$\#n + d \rightarrow d$	Add immediate to destination
ADDQ ⁴	BWL	#n,d	*****	d	d	d	d	d	d	d	d	d	-	-	-	s	$\#n + d \rightarrow d$	Add quick immediate (#n range: 1 to 8)
ADDX	BWL	Dy,Dx -(Ay),-(Ax)	*****	e	-	-	-	-	-	-	-	-	-	-	-	-	$Dy + Dx + X \rightarrow Dx$ $-(Ay) + -(Ax) + X \rightarrow -(Ax)$	Add source and eXtend bit to destination
AND ⁴	BWL	s,Dn Dn,d	---*00	e	-	s	s	s	s	s	s	s	s	s	s	s ⁴	$s \text{ AND } Dn \rightarrow Dn$ $Dn \text{ AND } d \rightarrow d$	Logical AND source to destination (ANDI is used when source is #n)
ANDI ⁴	BWL	#n,d	---*00	d	-	d	d	d	d	d	d	d	-	-	-	s	$\#n \text{ AND } d \rightarrow d$	Logical AND immediate to destination
ANDI ⁴	B	#n,CCR	=====	-	-	-	-	-	-	-	-	-	-	-	-	s	$\#n \text{ AND } CCR \rightarrow CCR$	Logical AND immediate to CCR
ANDI ⁴	W	#n,SR	=====	-	-	-	-	-	-	-	-	-	-	-	-	s	$\#n \text{ AND } SR \rightarrow SR$	Logical AND immediate to SR (Privileged)
ASL	BWL	Dx,Dy #n,Dy	*****	e	-	-	-	-	-	-	-	-	-	-	-	s		Arithmetic shift Dy by Dx bits left/right
ASR	W	d	-	-	-	d	d	d	d	d	d	d	-	-	-	s		Arithmetic shift Dy #n bits L/R (#n: 1 to 8)
	W	d	-	-	d	d	d	d	d	d	d	d	-	-	-	-		Arithmetic shift d 1 bit left/right (W only)
Bcc	BW ²	address ²	-----	-	-	-	-	-	-	-	-	-	-	-	-	-	if cc true then address \rightarrow PC	Branch conditionally (cc table on back) (8 or 16-bit + offset to address)
BCHG	B L	Dn,d #n,d	---*---	e ¹	-	d	d	d	d	d	d	d	-	-	-	s	$\text{NOT}(\text{bit number of } d) \rightarrow Z$ $\text{NOT}(\text{bit } n \text{ of } d) \rightarrow \text{bit } n \text{ of } d$	Set Z with state of specified bit in d then invert the bit in d
BCLR	B L	Dn,d #n,d	---*---	e ¹	-	d	d	d	d	d	d	d	-	-	-	s	$\text{NOT}(\text{bit number of } d) \rightarrow Z$ $0 \rightarrow \text{bit number of } d$	Set Z with state of specified bit in d then clear the bit in d
BFCHG	S	d(s.w)	---*00	d	-	d	-	-	-	d	d	d	-	-	-	-	$\text{NOT} \text{ bit field of } d$	Complement the bit field at destination
BFCLR	S	d(s.w)	---*00	d	-	d	-	-	-	d	d	d	-	-	-	-	$0 \rightarrow \text{bit field of } d$	Clear the bit field at destination
BFEXTS	S	s(s.w),Dn	---*00	d	-	s	-	-	-	s	s	s	s	s	s	-	bit field of s extend 32 \rightarrow Dn	Dn = bit field of s sign extended to 32 bits
BFEXTU	S	s(s.w),Dn	---*00	d	-	s	-	-	-	s	s	s	s	s	s	-	bit field of s unsigned \rightarrow Dn	Dn = bit field of s zero extended to 32 bits
BFFFD	S	s(s.w),Dn	---*00	d	-	s	-	-	-	s	s	s	s	s	s	-	bit number of 1 st 1 \rightarrow Dn	Dn = bit position of 1 st 1 or offset + width
BFINs	S	Dn,s(s.w)	---*00	s	-	d	-	-	-	d	d	d	-	-	-	-	low bits Dn \rightarrow bit field at d	Insert low bits of Dn to bit field at d
BFBSET	S	d(s.w)	---*00	d	-	d	-	-	-	d	d	d	-	-	-	-	$1 \rightarrow \text{bit field of } d$	Set all bits in bit field of destination
BFTST	S	d(s.w)	---*00	d	-	d	-	-	-	d	d	d	-	-	-	-	set CCR with bit field of d	N = high bit of bit field, Z set if all bits 0
BRA	BW ²	address ²	-----	-	-	-	-	-	-	-	-	-	-	-	-	-	address \rightarrow PC	Branch always (8 or 16-bit + offset to addr)
BSET	B L	Dn,d #n,d	---*---	e ¹	-	d	d	d	d	d	d	d	-	-	-	s	$\text{NOT}(\text{bit } n \text{ of } d) \rightarrow Z$ $1 \rightarrow \text{bit } n \text{ of } d$	Set Z with state of specified bit in d then set the bit in d
BSR	BW ²	address ²	-----	-	-	-	-	-	-	-	-	-	-	-	-	-	PC \rightarrow -(SP); address \rightarrow PC	Branch to subroutine (8 or 16-bit + offset)
BTST	B L	Dn,d #n,d	---*---	e ¹	-	d	d	d	d	d	d	d	-	-	-	s	$\text{NOT}(\text{bit } n \text{ of } d) \rightarrow Z$ $\text{NOT}(\text{bit } \#n \text{ of } d) \rightarrow Z$	Set Z with state of specified bit in d Leave the bit in d unchanged
CHK	W	s,Dn	---000	e	-	s	s	s	s	s	s	s	s	s	s	s	if $Dn < 0$ or $Dn > s$ then TRAP	Compare Dn with 0 and upper bound (s)
CLR	BWL	d	---000	d	-	d	d	d	d	d	d	d	-	-	-	-	$0 \rightarrow d$	Clear destination to zero
CMP ⁴	BWL	s,Dn	---***	e	s ⁴	s	s	s	s	s	s	s	s	s	s	s ⁴	set CCR with Dn - s	Compare Dn to source
CMFA ⁴	WL	s,An	---***	s	e	s	s	s	s	s	s	s	s	s	s	s	set CCR with An - s	Compare An to source
CMPI ⁴	BWL	#n,d	---***	d	-	d	d	d	d	d	d	d	-	-	-	s	set CCR with d - #n	Compare destination to #n
CMPM ⁴	BWL	(Ay),-(Ax)+	---***	-	-	-	-	-	-	-	-	-	-	-	-	-	set CCR with (Ax) - (Ay)	Compare (Ax) to (Ay); Increment Ax and Ay
DBcc	W	Dn,address ²	-----	-	-	-	-	-	-	-	-	-	-	-	-	-	if cc false then { Dn-1 \rightarrow Dn if $Dn < 0$ then addr \rightarrow PC }	Test condition, decrement and branch (16-bit + offset to address)
DIVS	W	s,Dn	---*0	e	-	s	s	s	s	s	s	s	s	s	s	s	$\pm 32\text{bit } Dn / \pm 16\text{bit } s \rightarrow \pm Dn$	Dn = (16-bit remainder, 16-bit quotient)
DIVU	W	s,Dn	---*0	e	-	s	s	s	s	s	s	s	s	s	s	s	$32\text{bit } Dn / 16\text{bit } s \rightarrow Dn$	Dn = (16-bit remainder, 16-bit quotient)
EOR ⁴	BWL	Dn,d	---*00	e	-	d	d	d	d	d	d	d	-	-	-	s ⁴	$Dn \text{ XOR } d \rightarrow d$	Logical exclusive OR Dn to destination
EORI ⁴	BWL	#n,d	---*00	d	-	d	d	d	d	d	d	d	-	-	-	s	$\#n \text{ XOR } d \rightarrow d$	Logical exclusive OR #n to destination
EORI ⁴	B	#n,CCR	=====	-	-	-	-	-	-	-	-	-	-	-	-	-	$\#n \text{ XOR } CCR \rightarrow CCR$	Logical exclusive OR #n to CCR
EORI ⁴	W	#n,SR	=====	-	-	-	-	-	-	-	-	-	-	-	-	-	$\#n \text{ XOR } SR \rightarrow SR$	Logical exclusive OR #n to SR (Privileged)
EXG	L	Rx,Ry	-----	e	e	-	-	-	-	-	-	-	-	-	-	-	register \leftrightarrow register	Exchange registers (32-bit only)
EXT	WL	Dn	---*00	d	-	-	-	-	-	-	-	-	-	-	-	-	$Dn.B \rightarrow Dn.W \mid Dn.W \rightarrow Dn.L$	Sign extend (change .B to .W or .W to .L)
ILLEGAL			-----	-	-	-	-	-	-	-	-	-	-	-	-	-	PC \rightarrow -(SSP); SR \rightarrow -(SSP)	Generate illegal instruction exception
JMP		d	-----	-	-	d	-	-	-	d	d	d	d	d	d	-	$\uparrow d \rightarrow PC$	Jump to effective address of destination
JSR		d	-----	-	-	d	-	-	-	d	d	d	d	d	d	-	PC \rightarrow -(SP); $\uparrow d \rightarrow PC$	push PC, jump to subroutine at address d
LEA	L	s,An	-----	-	e	s	-	-	-	s	s	s	s	s	s	-	$\uparrow s \rightarrow An$	Load effective address of s to An
LINK		An,#n	-----	-	-	-	-	-	-	-	-	-	-	-	-	-	An \rightarrow -(SP); SP \rightarrow An SP + #n \rightarrow SP	Create local workspace on stack (negative n to allocate space)
LSL	BWL	Dx,Dy #n,Dy	---*0*	e	-	-	-	-	-	-	-	-	-	-	-	-		Logical shift Dy, Dx bits left/right
LSR	W	d	-	-	-	d	d	d	d	d	d	d	-	-	-	s		Logical shift Dy, #n bits L/R (#n: 1 to 8)
	W	d	-	-	d	d	d	d	d	d	d	d	-	-	-	-		Logical shift d 1 bit left/right (W only)
MOVE ⁴	BWL	s,d	---*00	e	s ⁴	e	e	e	e	e	e	e	s	s	s	s ⁴	$s \rightarrow d$	Move data from source to destination
MOVE	W	s,CCR	=====	s	-	s	s	s	s	s	s	s	s	s	s	s	$s \rightarrow CCR$	Move source to Condition Code Register
MOVE	W	s,SR	=====	s	-	s	s	s	s	s	s	s	s	s	s	s	$s \rightarrow SR$	Move source to Status Register (Privileged)
MOVE	W	SR,d	-----	d	-	d	d	d	d	d	d	d	-	-	-	-	SR \rightarrow d	Move Status Register to destination
	BWL	s,d	XN2VC	Dn	An	(An)	(An)+	-(An)	(i,An)	(i,An,Rn)	abs.W	abs.L	(i,PC)	(i,PC,Rn)	#n			

Opcode	Size	Operand	CCR	Effective Address s=source, d=destination, e=either, i=displacement												Operation	Description	
	BWL	s,d	XNZVC	Dn	An	(An)	(An)+	-(An)	(i,An)	(i,An,Rn)	abs.W	abs.L	(i,PC)	(i,PC,Rn)	#n			
MOVE	L	USP,An An,USP	-----	-	d	-	-	-	-	-	-	-	-	-	-	USP → An An → USP	Move User Stack Pointer to An (Privileged) Move An to User Stack Pointer (Privileged)	
MOVEA ⁴	WL	s,An	-----	s	e	s	s	s	s	s	s	s	s	s	s	s → An	Move source to An (MOVE s,An use MOVEA)	
MOVEW ⁴	WL	Rn-Rn,d s,Rn-Rn	-----	-	-	d	-	d	d	d	d	d	-	-	-	Registers → d s → Registers	Move specified registers to/from memory (W source is sign-extended to L for Rn)	
MOVEB	WL	Dn,(i,An) (i,An),Dn	-----	s	-	-	-	-	d	-	-	-	-	-	-	Dn → (i,An)...(i+2,An)...(i+4,An) (i,An) → Dn...(i+2,An)...(i+4,An)	Move Dn to/from alternate memory bytes (Access only even or odd addresses)	
MOVEQ ⁴	L	#n,Dn	---*00	d	-	-	-	-	-	-	-	-	-	-	-	s #n → Dn	Move sign extended 8-bit #n to Dn	
MULS	W	s,Dn	---*00	e	-	s	s	s	s	s	s	s	s	s	s	s	16bit s * 16bit Dn → 32bit Dn	Multiply signed 16-bit; result: signed 32-bit
MULU	W	s,Dn	---*00	e	-	s	s	s	s	s	s	s	s	s	s	s	16bit s * 16bit Dn → Dn	Multiply unsig'd 16-bit; result: unsig'd 32-bit
NBCD	B	d	*U*U*	d	-	d	d	d	d	d	d	d	-	-	-	-	0 - d ₀ - X → d	Negate BCD with eXtend, BCD result
NEG	BWL	d	*****	d	-	d	d	d	d	d	d	d	-	-	-	-	0 - d → d	Negate destination (2's complement)
NEGX	BWL	d	*****	d	-	d	d	d	d	d	d	d	-	-	-	-	0 - d - X → d	Negate destination with eXtend
NOP			-----	-	-	-	-	-	-	-	-	-	-	-	-	-	None	No operation occurs
NOT	BWL	d	---*00	d	-	d	d	d	d	d	d	d	-	-	-	-	NOT(d) → d	Logical NOT destination (1's complement)
OR ⁴	BWL	s,Dn Dn,d	---*00	e	-	s	s	s	s	s	s	s	s	s	s	s ⁴	s OR Dn → Dn Dn OR d → d	Logical OR (ORI is used when source is #n)
ORI ⁴	BWL	#n,d	---*00	d	-	d	d	d	d	d	d	d	-	-	-	s	#n OR d → d	Logical OR #n to destination
ORI ⁴	B	#n,CCR	=====	-	-	-	-	-	-	-	-	-	-	-	-	s	#n OR CCR → CCR	Logical OR #n to CCR
ORI ⁴	W	#n,SR	=====	-	-	-	-	-	-	-	-	-	-	-	-	s	#n OR SR → SR	Logical OR #n to SR (Privileged)
PEA	L	s	-----	-	-	s	-	-	s	s	s	s	s	s	s	-	↑s → -(SP)	Push effective address of s onto stack
RESET			-----	-	-	-	-	-	-	-	-	-	-	-	-	-	Assert RESET Line	Issue a hardware RESET (Privileged)
ROL	BWL	Dx,Dy	---*0*	e	-	-	-	-	-	-	-	-	-	-	-	-		Rotate Dy, Dx bits left/right (without X)
ROR	W	#n,Dy		d	-	-	-	-	-	-	-	-	-	-	-	s		Rotate Dy, #n bits left/right (#n: 1 to 8) Rotate d 1-bit left/right (W only)
ROXL	BWL	Dx,Dy	***0*	e	-	-	-	-	-	-	-	-	-	-	-	-		Rotate Dy, Dx bits L/R. X used then updated
ROXR	W	#n,Dy		d	-	-	-	-	-	-	-	-	-	-	-	s		Rotate Dy, #n bits left/right (#n: 1 to 8) Rotate destination 1-bit left/right (W only)
RTE			=====	-	-	-	-	-	-	-	-	-	-	-	-	-	(SP)+ → SR; (SP)+ → PC	Return from exception (Privileged)
RTR			=====	-	-	-	-	-	-	-	-	-	-	-	-	-	(SP)+ → CCR; (SP)+ → PC	Return from subroutine and restore CCR
RTS			-----	-	-	-	-	-	-	-	-	-	-	-	-	-	(SP)+ → PC	Return from subroutine
SBCC	B	Dy,Dx -(Ay),-(Ax)	*U*U*	e	-	-	-	-	-	-	-	-	-	-	-	-	Dx ₀ - Dy ₀ - X → Dx ₀ -(Ax) ₀ - (Ay) ₀ - X → -(Ax) ₀	BCD destination - BCD source - eXtend Z cleared if result not 0 unchanged otherwise
SCC	B	d	-----	d	-	d	d	d	d	d	d	d	-	-	-	-	If cc is true then 1's → d else 0's → d	If cc true then d.B = 11111111 else d.B = 00000000
STOP		#n	=====	-	-	-	-	-	-	-	-	-	-	-	-	s	#n → SR; STOP	Move #n to SR, stop processor (Privileged)
SUB ⁴	BWL	s,Dn Dn,d	*****	e	s	s	s	s	s	s	s	s	s	s	s	s ⁴	Dn - s → Dn d - Dn → d	Subtract binary (SUBI or SUBQ used when source is #n. Prevent SUBQ with #n.L)
SUBA ⁴	WL	s,An	-----	s	e	s	s	s	s	s	s	s	s	s	s	s	An - s → An	Subtract address (W sign-extended to L)
SUBI ⁴	BWL	#n,d	*****	d	-	d	d	d	d	d	d	d	-	-	-	s	d - #n → d	Subtract immediate from destination
SUBQ ⁴	BWL	#n,d	*****	d	-	d	d	d	d	d	d	d	-	-	-	s	d - #n → d	Subtract quick immediate (#n range: 1 to 8)
SUBX	BWL	Dy,Dx -(Ay),-(Ax)	*****	e	-	-	-	-	-	-	-	-	-	-	-	-	Dx - Dy - X → Dx -(Ax) - (Ay) - X → -(Ax)	Subtract source and eXtend bit from destination
SWAP	W	Dn	---*00	d	-	-	-	-	-	-	-	-	-	-	-	-	bits[31:16] ↔ bits[15:0]	Exchange the 16-bit halves of Dn
TAS	B	d	---*00	d	-	d	d	d	d	d	d	d	-	-	-	-	test d → CCR; 1 → bit7 of d	N and Z set to reflect d, bit7 of d set to 1
TRAP		#n	-----	-	-	-	-	-	-	-	-	-	-	-	-	s	PC → -(SSP); SR → -(SSP); (vector table entry) → PC	Push PC and SR. PC set by vector table #n (#n range: 0 to 15)
TRAPV			-----	-	-	-	-	-	-	-	-	-	-	-	-	-	If V then TRAP #7	If overflow, execute an Overflow TRAP
TST	BWL	d	---*00	d	-	d	d	d	d	d	d	d	-	-	-	-	test d → CCR	N and Z set to reflect destination
UNLK		An	-----	-	d	-	-	-	-	-	-	-	-	-	-	-	An → SP; (SP)+ → An	Remove local workspace from stack
	BWL	s,d	XNZVC	Dn	An	(An)	(An)+	-(An)	(i,An)	(i,An,Rn)	abs.W	abs.L	(i,PC)	(i,PC,Rn)	#n			

Condition Tests (* OR, ! NOT, ⊕ XOR, * Unsigned, * Alternate cc)					
cc	Condition	Test	cc	Condition	Test
T	true	I	VC	overflow clear	IV
F	false	O	VS	overflow set	V
HI*	higher than	(C + Z)	PL	plus	IN
LS*	lower or same	C + Z	MI	minus	N
HS*, CC*	higher or same	!C	GE	greater or equal	!(N ⊕ V)
LP*, CS*	lower than	C	LT	less than	(N ⊕ V)
NE	not equal	!Z	GT	greater than	!(N ⊕ V) + Z
EQ	equal	Z	LE	less or equal	(N ⊕ V) + Z

An Address register (16/32-bit, n=0-7)
Dn Data register (8/16/32-bit, n=0-7)
Rn any data or address register
BCD Binary Coded Decimal
PC Program Counter (24-bit)
#n Immediate data
SP Active Stack Pointer (same as A7)
¹ Long only; all others are byte only
² Branch sizes: **B** or **S** -128 to +127 bytes, **W** or **L** -32768 to +32767 bytes
³ Assembler automatically uses A, L, Q or M form if possible. Use #n.L to prevent Quick optimization
⁴ Bit field determines size. Not supported by 68000. EASy68K hybrid form of 68020 instruction

s Source,
d Destination
e Either source or destination
i Displacement
↑ Effective address
{ow} offset: width of bit field
SSP Supervisor Stack Pointer (32-bit)
USP User Stack Pointer (32-bit)
SR Status Register (16-bit)
CCR Condition Code Register (lower 8-bits of SR)
N negative, **Z** zero, **V** overflow, **C** carry, **X** extend
***** set by operation's result, **=** set directly
- not affected, **O** cleared, **I** set, **U** undefined

Distributed under GNU general public use license

Commonly Used Simulator Input/Output Tasks						TRAP #15 is used to run simulator tasks. Place the task number in register D0. See Help for a complete description of available tasks. (cstring is null terminated)					
0	Display n characters of string at (A1), n=D1.W (stops on NULL or max 255) with CR,LF	1	Display n characters of string at (A1), n=D1.W (stops on NULL or max 255) without CR,LF	2	Read characters from keyboard. Store at (A1). Null terminated. D1.W = length (max 80)	3	Display D1.L as signed decimal number				
4	Read number from keyboard into D1.L	5	Read single character from keyboard in D1.B	6	Display D1.B as ASCII character	7	Set D1.B to 1 if keyboard input pending else set to 0				
8	time in 1/100 second since midnight → D1.L	9	Terminate the program. (Halts the simulator)	10	Print cstring at (A1) on default printer.	11	Position cursor at row,col D1.W=ccrr. \$FFD0 clears				
13	Display cstring at (A1) with CR,LF	14	Display cstring at (A1) without CR,LF	15	Display unsigned number in D1.L in D2.B base	17	Display cstring at (A1), then display number in D1.L				
18	Display cstring at (A1), read number into D1.L	19	Return state of keys or scan code. See help	20	Display * number in D1.L, field D2.B columns wide	21	Set font properties. See help for details				

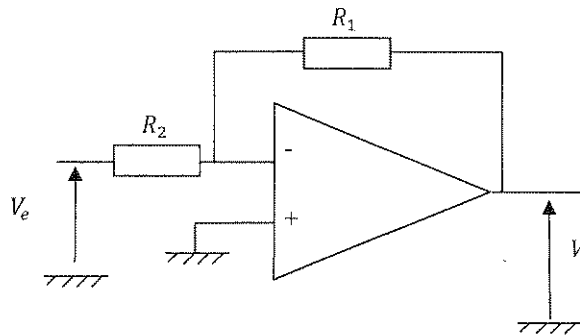
NOM : PRENOM : GROUPE :

Contrôle 2 Electronique

*Les calculatrices et les documents ne sont pas autorisés. Le barème est donné à titre indicatif.
Réponses exclusivement sur le sujet*

Partie A. Amplificateur opérationnel (3 points)

Soit le circuit suivant :

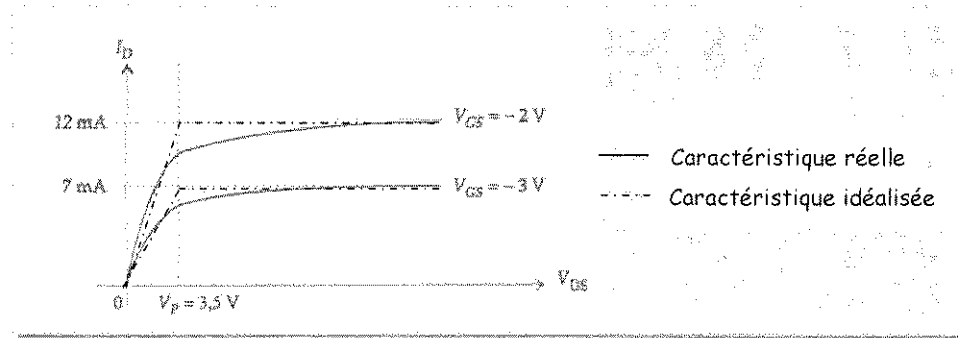


a) L'AOP fonctionne-t-il en mode linéaire? Pourquoi?

b) Déterminer l'expression de V_s en fonction de V_e et des résistances.

Partie B. Transistors à effet de champ - Polarisation (9 points)Exercice 1. (2 points)

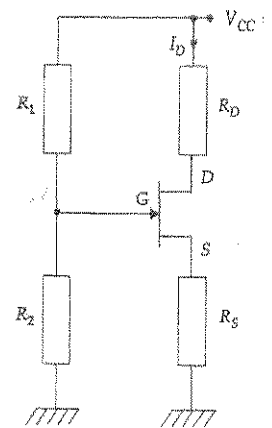
On considère un transistor à effet de champ à jonction canal N, et son réseau de caractéristiques présenté sur le graphique suivant :



Rq : Pour toute utilisation du graphique, travaillez avec les caractéristiques idéalisées.

On l'inclut dans le montage ci-contre.

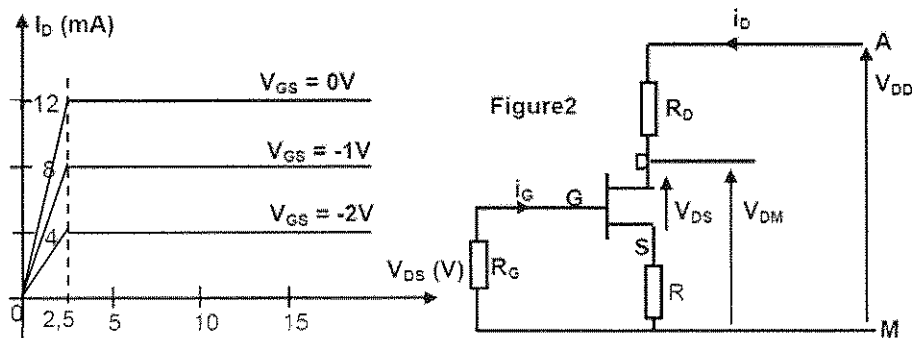
Déterminer la condition sur la valeur de la résistance ainsi que la valeur de pour que le transistor soit polarisé dans sa zone de fonctionnement linéaire avec



Exercice 2. (7 points)

On considère un Transistor à Effet de Champ dont les caractéristiques sont données dans la figure 1. Ce TEC est utilisé dans le montage figure 2. On donne :

- ✓ Tension d'alimentation $V_{DD} = 15V$
- ✓ Le point de fonctionnement est choisi tel que la tension $V_{DM} = 7V$.



Dans un premier temps, on veut que le transistor fonctionne dans sa zone linéaire.

- a) Calculer l'intensité du courant I_D sachant que $R_D = 1k\Omega$ et en déduire la tension V_{GS} .

- b) Déterminer la valeur de la résistance R

c) Le transistor fonctionne-t-il bien dans sa zone linéaire? Pourquoi?

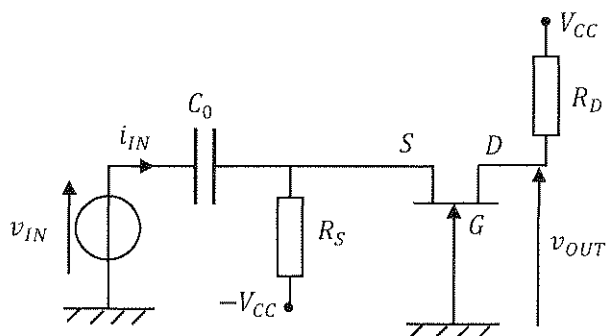
On veut maintenant travailler dans la zone ohmique, avec $V_{GS} = -2V$.

d) A quoi est équivalent le canal Drain-Source dans ce mode de fonctionnement? Donner les caractéristiques de cet équivalent.

e) Déterminer la résistance R_D pour avoir ce type de fonctionnement, sachant que $R = 1k\Omega$.

Partie C. Transistors à effet de champ - Petits signaux (3 points)

Dans le schéma ci-dessous, le transistor à effet de champ est monté en grille commune.

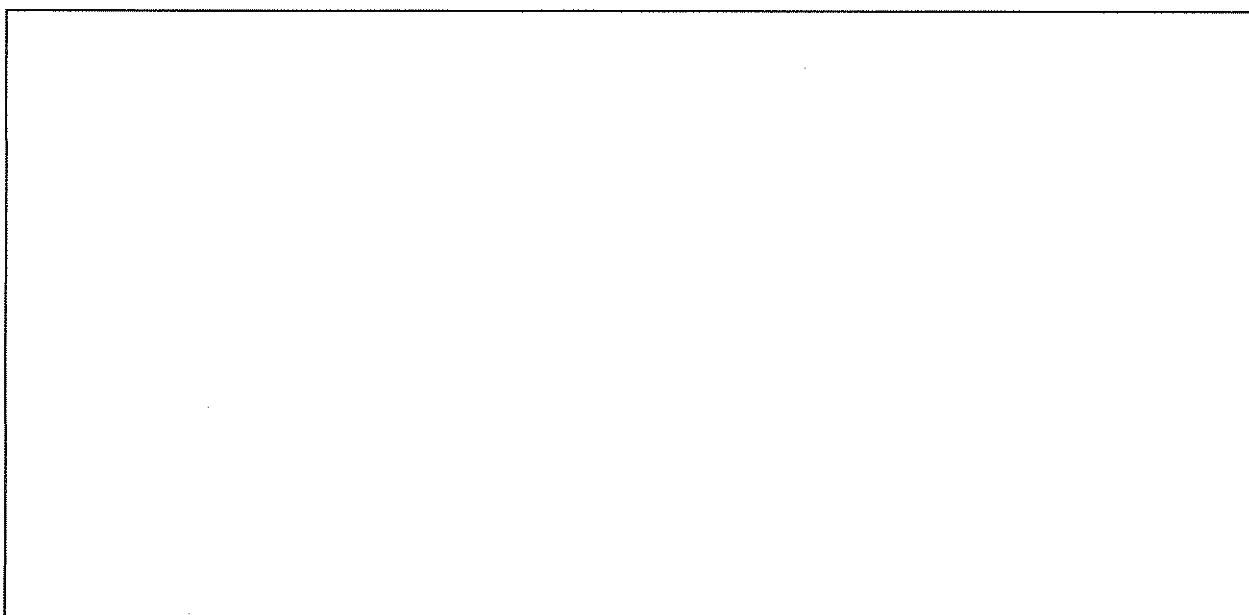


a) Dessiner le schéma équivalent petits signaux du montage.



b) Déterminer l'amplification en tension de ce montage.

Rq : On pourra exprimer les tensions d'entrée et de sortie en fonction de v_{gs} .

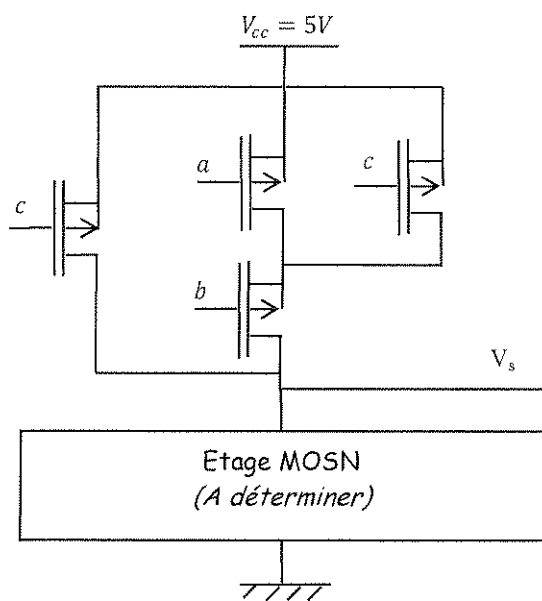


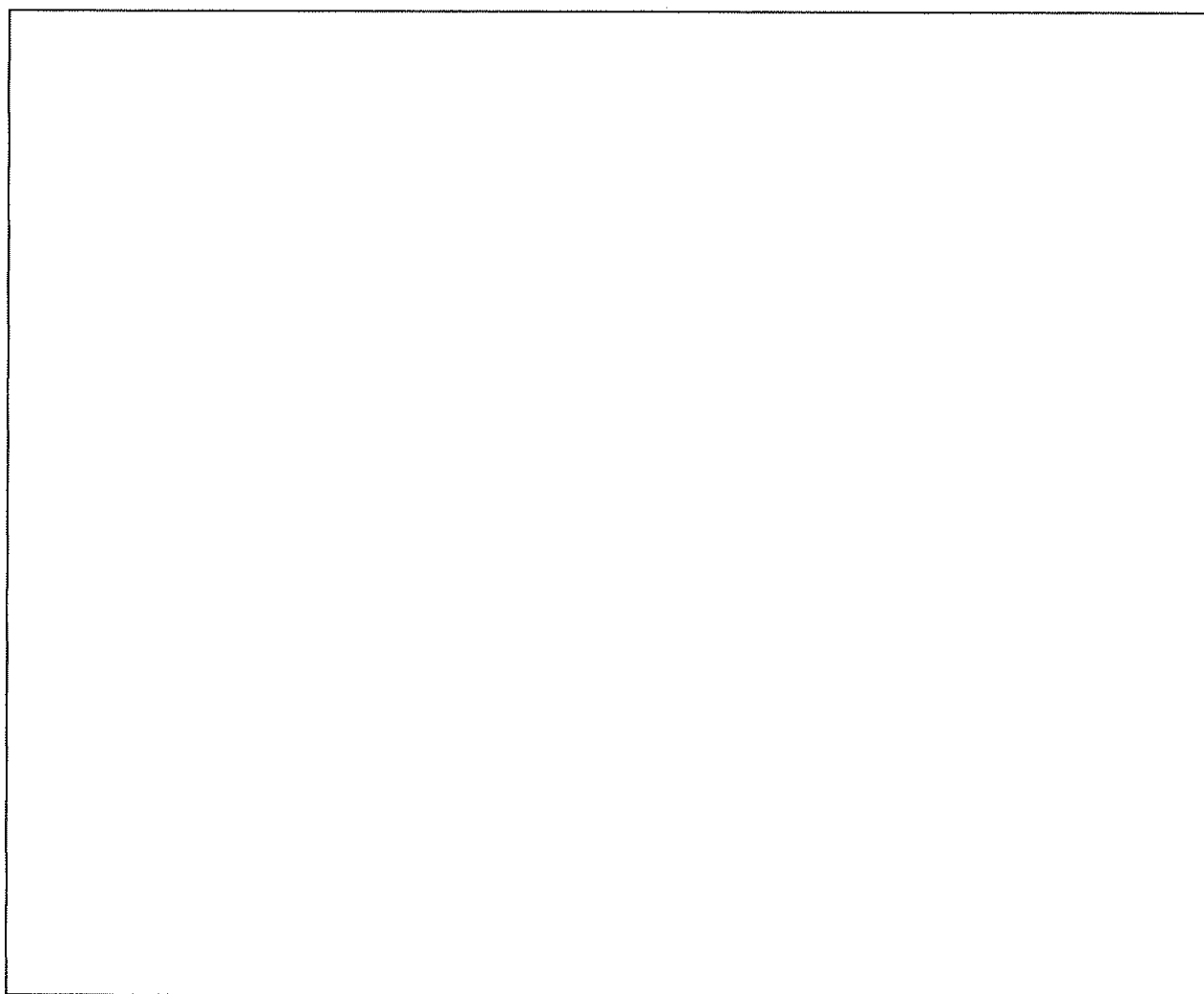
Partie D. Transistors MOS et Portes Logiques (5 points)

- a) Rappeler les conditions de passage (canal Drain Source conducteur) et de blocage (canal Drain Source non conducteur) pour les MOSFET Canal N et P.

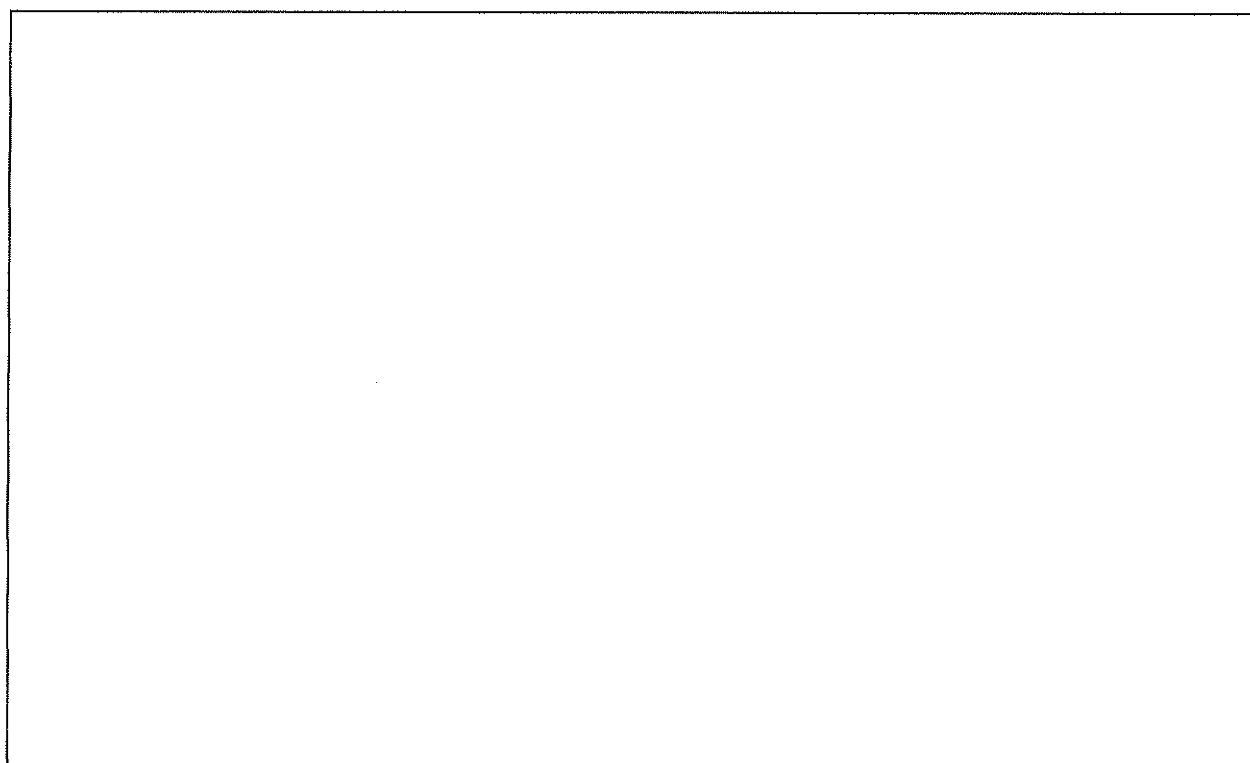
- b) Soit le montage suivant : Il correspond à une fonction logique réalisée en technologie CMOS.

Déterminer l'équation logique correspondant à cette fonction (justifiez votre réponse), puis, après l'avoir simplifiée (l'équation (!)), redessiner le schéma COMPLET de la porte logique, y compris l'étage MOSN, non dessiné sur le montage initial.





Si vous manquez de place, vous pouvez utiliser le cadre ci-dessous.



Contrôle n° 2 de Physique
Documents et calculatrice non autorisés

Partie cours

(7 points)

Les questions sont indépendantes.

- 1- **a-** Interpréter l'expérience de **Franck-Hertz**, en précisant le but et le principe de cette expérience.
b- Préciser l'étude expérimentale faite en parallèle pour valider l'interprétation de cette expérience. Justifier votre réponse
- 2- **a-** Donner la définition de τ : **durée de vie** d'un état d'énergie excité.
b- Comment intervient cette durée de vie dans l'expression de la population de l'état excité : $N(t)$
c- Citer un autre phénomène physique où apparaît le même type de variation $N(t)$
- 4- A l'aide du **modèle de Bohr**, on montre que l'énergie totale de l'électron en fonction du nombre quantique principal n est :
$$E_n = -\frac{13.6}{n^2} \quad (\text{Exprimée en eV})$$

a- Utiliser cette dernière expression et un des postulats de Bohr pour retrouver la fréquence de transition (de m vers n où $m > n$) donnée par : $\nu_{m,n} = A\left(\frac{1}{n^2} - \frac{1}{m^2}\right)$.
Préciser l'expression de la constante A .
b- En déduire la longueur d'onde de la transition : $\frac{1}{\lambda_{m,n}} = R_H\left(\frac{1}{n^2} - \frac{1}{m^2}\right)$.
Préciser l'expression de la **constante de Rydberg** R_H .
- 5- **a-** Interpréter la courbe de Moseley en précisant la conséquence
b- Citer les trois cas où l'on peut appliquer le modèle de Bohr
c- Donner pour chacun des cas, l'expression de l'énergie d'un niveau de nombre quantique principal n en précisant la correction à apporter.

Exercice

Partie A

(6 points)

On considère un milieu diélectrique (isolant), composé de n électrons. L'électron de masse m est soumis à une force de rappel d'expression $\vec{f} = -K\vec{r}$; Où \vec{r} est le vecteur position de l'électron et K la constante de rappel.
On envoie dans le diélectrique une O.E.M.P.S de pulsation ω . On suppose pour ce milieu $\mu \approx \mu_0$ et $\varepsilon \approx \varepsilon_0$

- 1) a- Donner le P.F.D pour un électron, sachant que $F_e \gg F_m$, le vecteur accélération et le vecteur vitesse vérifient : $\vec{v} = \frac{d\vec{r}}{dt} \approx \frac{\partial \vec{r}}{\partial t} = -i\omega \cdot \vec{r}$ et $\vec{a} = \frac{d\vec{v}}{dt} \approx \frac{\partial \vec{v}}{\partial t} = -i\omega \cdot \vec{v}$
- b- Exprimer la position \vec{r} en fonction du champ électrique de l'onde, de ω et de ω_0 . Où $\omega_0^2 = K/m_e$, pulsation propre de l'oscillateur qui est l'électron.
En déduire l'expression du vecteur vitesse \vec{v}
- 2) Donner l'expression de la densité de courant \vec{J} en fonction du champ \vec{E} , en déduire la conductivité du milieu diélectrique γ .
- 3) a- Montrer que l'équation de dispersion peut se mettre sous la forme :

$$k^2 = \frac{\omega^2}{c^2} \left(\frac{\omega^2 - \Omega^2}{\omega^2 - \omega_0^2} \right). \text{ Où } \omega_p^2 = n_e \cdot e^2 / m_e \cdot \epsilon_0$$

- b- Identifier la pulsation Ω . Préciser les fréquences de coupure
c- Donner les types d'ondes dans ce milieu.

Partie B (4 points)

On considère maintenant le milieu plasma où les électrons sont libres et ne sont soumis qu'à la force électrique \vec{F}_e . L'onde électromagnétique (considérée comme OPPS) est envoyée dans ce milieu (où $\epsilon \approx \epsilon_0$ et $\mu \approx \mu_0$) avec une pulsation ω .

- 1- Donner sans refaire de calcul, la nouvelle expression de l'équation de dispersion, en utilisant celle obtenue pour le diélectrique (**Partie A**). Justifier votre réponse.
- 2- a- Préciser la nouvelle fréquence de coupure, ainsi que les types d'onde selon la fréquence.
b- Calculer la fréquence de coupure pour $n = 4.10^{12} m^{-3}$, $e = 1.6.10^{-19} C$, $\epsilon_0 = 9.10^{-12} S.I$,
 $m = 9.10^{-31} kg$

Partie C (3 points)

On envoie l'onde électromagnétique dans le même milieu plasma mais cette fois ci on ne néglige pas le mouvement des ions de densité n_i et de masse m_i . (Sachant que $n_e = n_i = n$).

- 1- On montre que la vitesse d'un électron est : $\vec{V}_e = -\frac{ie\vec{E}}{m_e\omega}$

Donner (sans refaire de calcul) la vitesse \vec{V}_i d'un ion de charge $+e$ et de masse m_i .

- 2- Donner l'expression de la densité de courant totale, sachant que : $\vec{J} = \vec{J}_e + \vec{J}_i$, en déduire la conductivité γ du milieu.
- 3- Donner la nouvelle expression de l'équation de dispersion ainsi que la nouvelle fréquence de coupure. On peut poser $\frac{1}{M} = \frac{1}{m_e} + \frac{1}{m_i}$

Formulaire

1) Force électrique

$$\vec{F}_e = q \cdot \vec{E}$$

2) Densité de courant pour une seule espèce de charge q

$$\vec{J} = n \cdot q \cdot \vec{V}$$

3) Loi d'Ohm généralisée

$$\vec{J} = \gamma \cdot \vec{E}$$

4) Equation de dispersion pour un milieu matériel quelconque

$$k^2 = \frac{\omega^2}{c^2} \left(1 + \frac{i\gamma}{\omega \cdot \epsilon_0} \right)$$

Algorithmique

Contrôle n° 2

INFO-SPÉ – API
EPITA

D.S. 311181.46 BW (7 mar 2012 - 10 :00)

Consignes (à lire) :

- ☐ Vous devez répondre sur les feuilles de réponses prévues à cet effet.
 - Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
 - Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées** : utilisez des brouillons !
 - Ne séparez pas les feuilles à moins de pouvoir les ré-agrafer pour les rendre.
 - Aucune réponse au crayon de papier ne sera corrigée.
 - ☐ La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
 - ☐ **Les algorithmes :**
 - Tout algorithme doit être écrit dans le langage ALGO (pas de C, CAML ou autre).
 - Tout code ALGO non indenté ne sera pas corrigé.
 - Tout ce dont vous avez besoin (types, routines) est indiqué dans l'énoncé.
 - ☐ Durée : 2h00
-



Exercice 1 (Graphes : dessiner c'est gagner – 4 points)

Soit le graphe $G = \langle S, A \rangle$ orienté avec :

$S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
et $A = \{(1, 2), (1, 3), (1, 5), (1, 6), (2, 3), (3, 4), (4, 9), (5, 2), (5, 8), (6, 7), (7, 1), (8, 3), (8, 4), (8, 9), (8, 10), (10, 9)\}$

1. Représenter graphiquement le graphe correspondant à G .
2. Donner le tableau `DemiDegréIntérieur` tel que $\forall i \in [1, \text{Card}(S)]$, `DemiDegréIntérieur[i]` soit égal au demi-degré intérieur de i dans G .
3. Représenter (dessiner) la forêt couvrante associée au parcours en profondeur du graphe G . *Ajouter aussi les autres arcs en les qualifiant à l'aide d'une légende explicite.* On considérera le sommet 1 comme base du parcours, les sommets devant être choisis en ordre numérique croissant.

Exercice 2 (Graphe sans circuit... (3 points))

1. Quelle est, au niveau de la classification de ses arcs, la particularité d'un graphe sans circuit ?
2. Soit $G = \langle S, A \rangle$ un graphe sans circuit, soient les tableaux *os* et *op* contenant, respectivement, les numéros d'ordre suffixe et préfixe de tous les sommets du graphe G obtenus lors du parcours en profondeur de G . Démontrer que pour une paire quelconque de sommets distincts $u, v \in S$, s'il existe un arc dans G de u à v , alors $os[v] < os[u]$.

Pour l'exercice suivant

Représentation statique des graphes

```
constantes
    MAX = /* une valeur suffisante ! */

types
    t_mat_adj  = MAX * MAX entier

    t_graphe_s = enregistrement
        entier    ordre
        booleen   orient
        t_mat_adj adj
    fin enregistrement

    t_vect_entiers = MAX entier
```

Exercice 3 (I want to be a tree – 13 pts)

Définition :

Un arbre est un graphe connexe sans cycle.

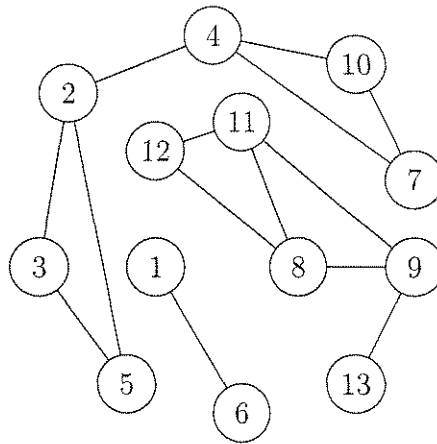


FIGURE 1 – Not a tree yet

Le but de cet exercice est de transformer un graphe en arbre en le modifiant le moins possible, à l'aide d'un parcours profond.

1. On effectue un parcours profond du graphe :
 - (a) Quelles sont les arêtes qui peuvent être enlevées du graphe sans augmenter le nombre de composantes connexes ?
 - (b) Comment repérer ces arêtes lors du parcours profond ?
 - (c) Appliquer au graphe de la figure 1 : donner la liste des arêtes que l'on supprimera lors du parcours profond en choisissant les sommets dans l'ordre croissant (y compris pour les successeurs).
2. Pendant le parcours profond, on attribue à chaque sommet un numéro de composante connexe (de 1 à k , s'il y a k composantes) :
 - (a) Combien d'arêtes faut-il ajouter pour rendre le graphe connexe ?
 - (b) Comment, lors du parcours, savoir quelles arêtes ajouter pour rendre le graphe connexe ?
 - (c) Donner le tableau des composantes connexes du graphe de la figure 1 obtenu lors du parcours profond toujours en choisissant les sommets dans l'ordre croissant.
3. **L'algorithme :**

L'algorithme demandé ici sera un parcours profond d'un graphe non orienté en représentation statique.

L'algorithme devra :

 - Construire le vecteur des composantes connexes, dans le vecteur cc .
 - Ajouter au graphe les arêtes qui permettent de le rendre connexe.
 - Supprimer du graphe les arêtes "inutiles" : sans augmenter le nombre de composantes.



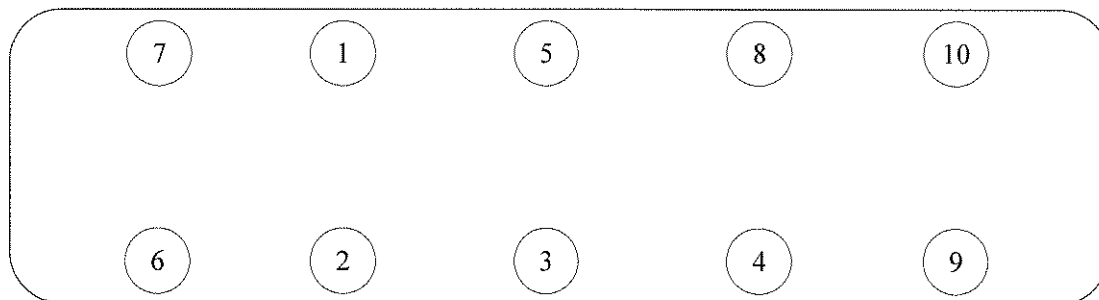
Nom	
Prénom	
Groupe	

Note	
------	--

Algorithmique - Info-SPÉ – API
 Contrôle n° 2
D.S. 311181.46 BW (7 mar. 2012 - 10 :00)
 Feuilles de réponses

Réponses 1 (Graphes : dessiner c'est gagner – 4 points)

1. Représenter le graphe correspondant à G.



2. Demi-degrés intérieurs de tous les sommets du graphe G :

	1	2	3	4	5	6	7	8	9	10
DemiDegréIntérieur										

3. Forêt couvrante :

Réponses 2 Graphe sans circuit... (3 points)

1. Quelle est, au niveau de la classification de ses arcs, la particularité d'un graphe sans circuit ?

2. Démontrer que pour une paire quelconque de sommets distincts $u, v \in S$, s'il existe un arc dans G de u à v , alors $os[v] < os[u]$.

Réponses 3 (I want to be a tree – 13 pts)

1. (a) Les arêtes qui peuvent être enlevées :

- (b) Comment repérer ces arêtes lors du parcours profondeur ?

- (c) La liste des arêtes du graphe "Not a tree yet" supprimées :

2. Pendant le parcours profondeur, on attribue à chaque sommet un numéro de composante connexe (de 1 à k , s'il y a k composantes) :

- (a) Nombre d'arêtes à ajouter :

- (b) Comment, lors du parcours, savoir quelles arêtes ajouter ?

- (c) Le tableau des composantes connexes du graphe "Not a tree yet" :

	1	2	3	4	5	6	7	8	9	10	11	12
cc												

3. L'algorithme d'appel :

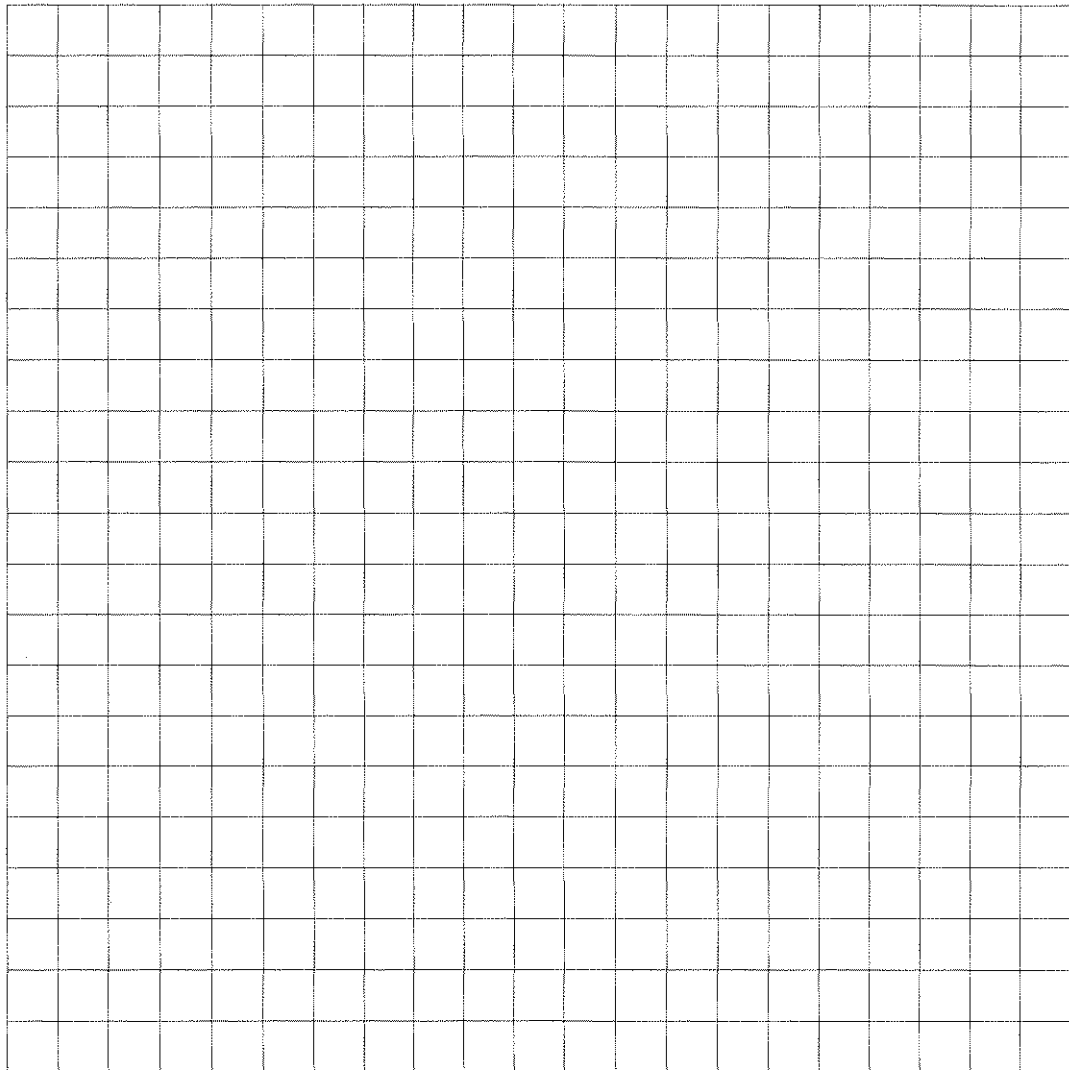
La procédure `make_me_tree` (`t_graphe_s` G , `t_vect_entiers` cc) transforme G en arbre et remplit cc , vecteur des composantes connexes du graphe de départ.

Elle utilise la procédure `prof_rec` (voir page suivante).

```
algorithme procedure make_me_tree
parametres globaux
    t_graphe_s      G
    t_vect_entiers   cc
```

variables

debut



```
fin algorithme procedure make_me_tree
```

L'algorithme récursif :

La procédure `prof_rec` (`t_graphe_s` G , entier s , $pere$, no_cc , `t_vect_entiers` cc) effectue le parcours profondeur du graphe G à partir du sommet s . $pere$ est le sommet père de s dans la forêt couvrante, no_cc est le numéro de la composante connexe actuelle et cc , qui sert de marque, est le vecteur de composantes.

algorithme procedure `prof_rec`

parametres locaux

`t_graphe_s` G

entier s , $pere$

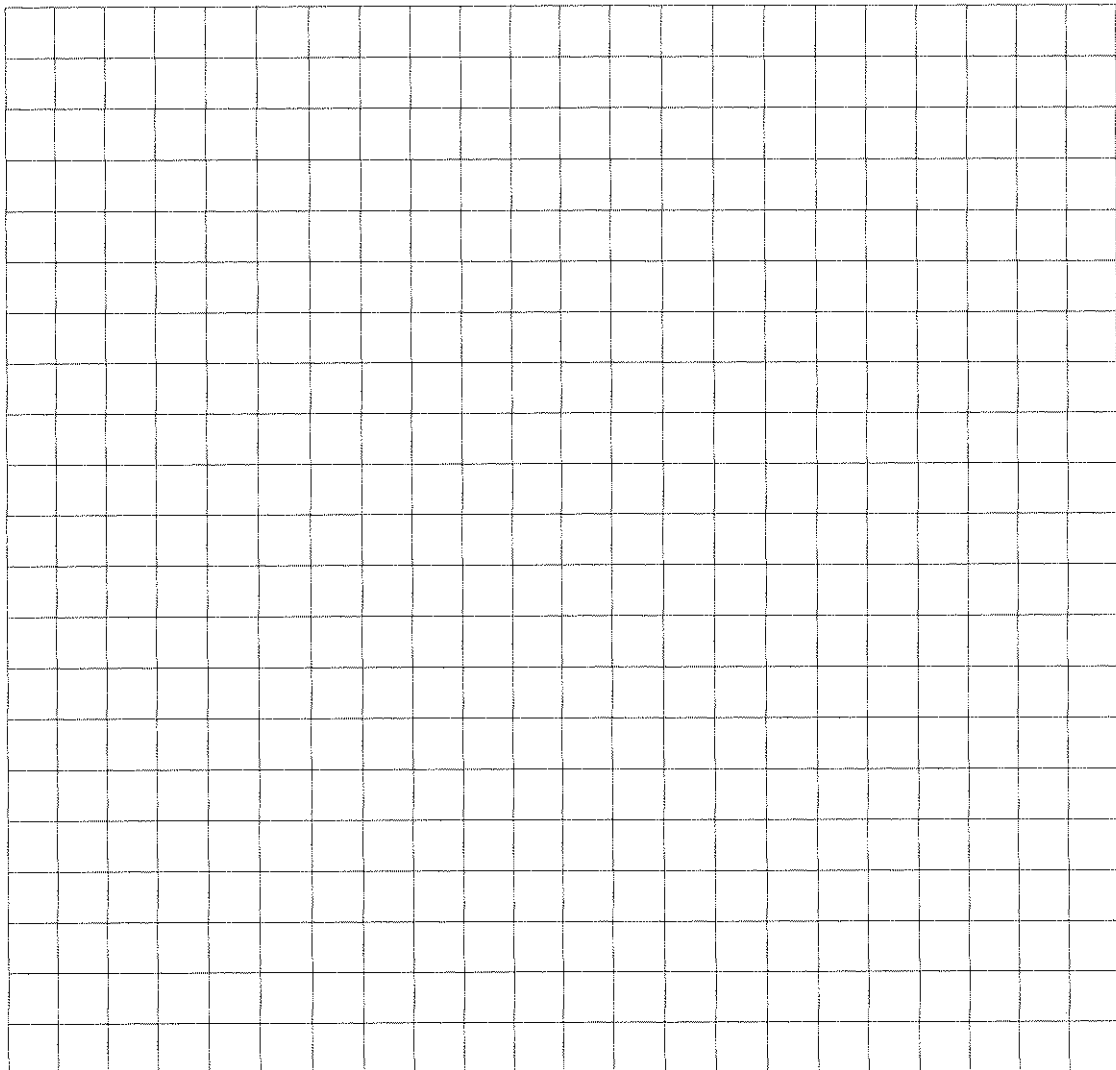
entier no_cc

parametres globaux

`t_vect_entiers` cc

variables

debut



fin algorithme procedure `prof_rec`

Contrôle 2

Durée : trois heures
Documents et calculatrices non autorisés

Nom :

Prénom :

Groupe :

Entourer le nom de votre professeur de TD : M. Ghanem/ Mme Malek

Consignes :

- vous devez répondre directement sur les feuilles jointes.
- *aucune autre feuille, que celles agrafées fournies pour répondre, ne sera corrigée.*

Exercice 1 (5 points)

Déterminer la nature des intégrales impropres suivantes :

1. $\int_0^{+\infty} \frac{t^2}{1+t^4} dt$

2. $\int_1^{+\infty} \ln\left(1 + \frac{1}{t^2}\right) dt$

3. $\int_0^{+\infty} \frac{e^t}{1+e^{2t}} dt$

4. $\int_0^{+\infty} \frac{\sin(t)}{t\sqrt{t}} dt$

Exercice 2 (4 points)

Soient $n \in \mathbb{N}$ et $I_n = \int_0^{+\infty} x^{2n+1} e^{-x^2} dx$.

1. Déterminer la nature de I_n en fonction de n .

2. Déterminer I_0 et I_1 .

3. Via une intégration par parties, en posant $u(x) = x^{2n}$, exprimer I_n en fonction de I_{n-1} .

4. En déduire (sans récurrence) I_n en fonction de n .

Exercice 3 (2 points)

Soient $a \in \mathbb{R}$ et $I_a = \int_2^{+\infty} \frac{(\ln(t))^a}{t} dt$

1. Via le changement de variable $u = \ln(t)$, déterminer la nature de I_a en fonction de a .

2. Calculer I_a en fonction de a .

Exercice 4 (4,5 points)

Soient $E = \mathbb{R}_2[X]$ et $\Phi : E \times E \longrightarrow \mathbb{R}$ définie pour tout $(P, Q) \in E^2$ par

$$\Phi(P, Q) = P(0)Q(0) + P(1)Q(1) + P(2)Q(2)$$

1. Montrer que Φ est un produit scalaire sur E .

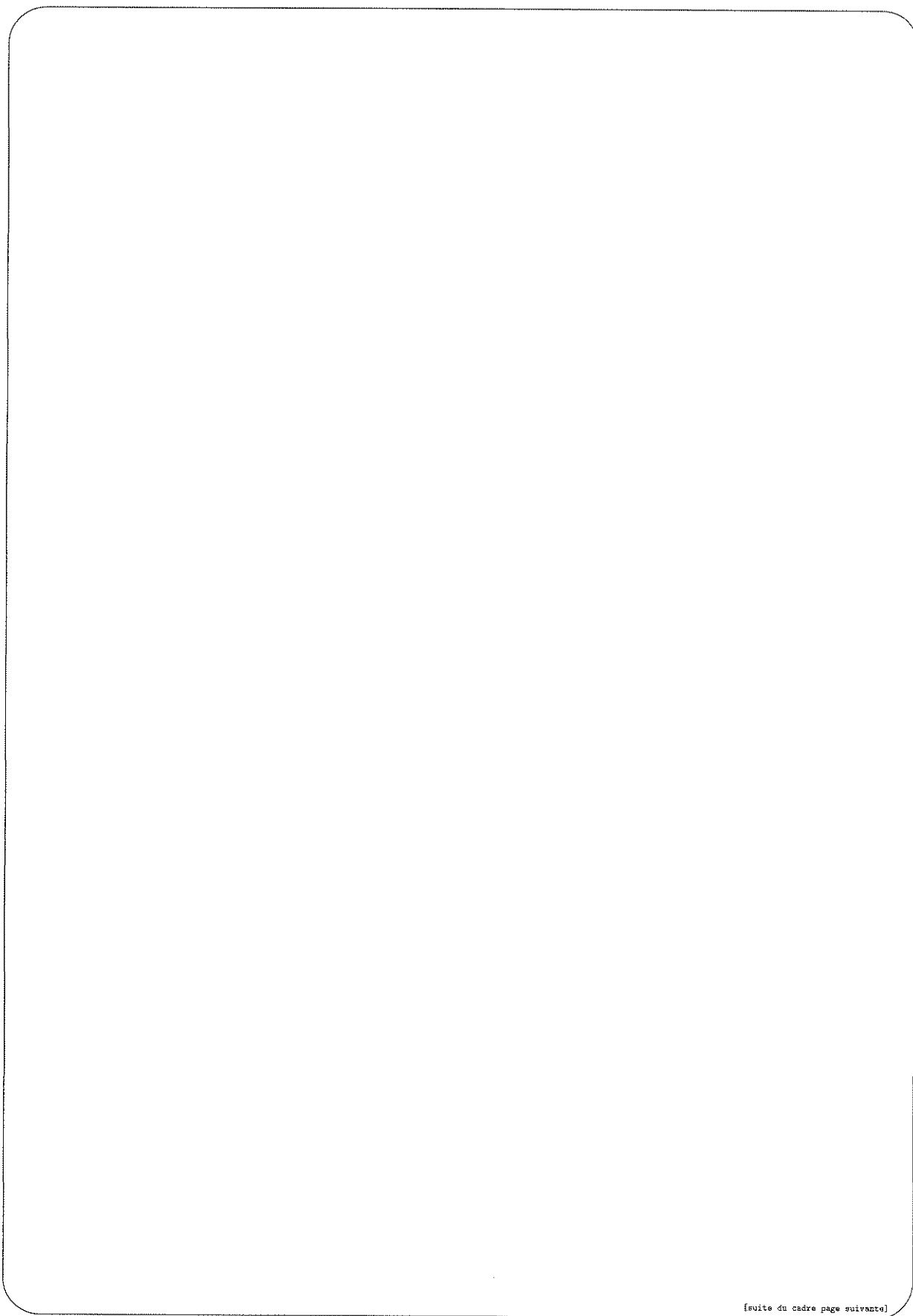
2. On note à présent Φ par \langle, \rangle de sorte que pour tout $(P, Q) \in E^2$,

$$\langle P, Q \rangle = P(0)Q(0) + P(1)Q(1) + P(2)Q(2)$$

Déterminer (par la méthode de Gram-Schmidt) à partir de la base canonique $(1, X, X^2)$ de E une base orthogonale (P_0, P_1, P_2) de (E, \langle, \rangle) .

N.B. : vous devez détailler la méthode de Gram-Schmidt pas à pas et non pas parachuter des formules donnant les coefficients inconnus.

[suite du cadre page suivante]



[suite du cadre page suivante]

Exercice 5 (3 points)

Soient (E, \langle, \rangle) un espace euclidien, $f \in \mathcal{L}(E)$ telle que $\forall (x, y) \in E^2 : \langle f(x), y \rangle = -\langle x, f(y) \rangle$ et $s = f \circ f$.

1. Montrer que $\text{Ker}(f) \perp \text{Im}(f)$.

2. Montrer que $\forall (x, y) \in E^2 : \langle s(x), y \rangle = \langle x, s(y) \rangle$.

3. Montrer que $\text{Sp}(s) \subset \mathbb{R}^-$ où $\text{Sp}(s)$ désigne l'ensemble des valeurs propres réelles de s .

Exercice 6 (2 points)

Soient (E, \langle, \rangle) euclidien, F et G deux sev de E tels que $E = F \oplus G$.

Montrer que $E = F^\perp \oplus G^\perp$.

[suite du cadre page suivante]

