Nom i Cognoms:_____ **Una possible solució**_____

1) Escriu un avantatge d'un sistema d'E/S mapejat a memòria i un avantatge d'un NO mapejat.

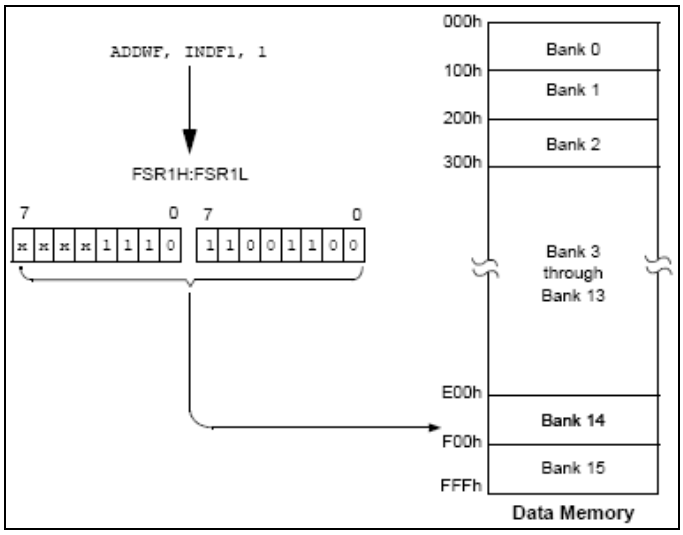
MAPEJAT:

Joc d'instucions mes senzill, menys senyals, arquitectura mes simple

NO MAPEJAT:

Amplia la memòria, facilita la gestió d'aquesta (no cal tenir espais reservats per a i/o), codis mes clars (in/out fan evident que estem accedint a i/o)

2) Per quin motiu calen 2 registres (FSR1H:FSR1L) per a fer adreçament indirecte, mentre que fent-ho amb accés directe només en cal un?

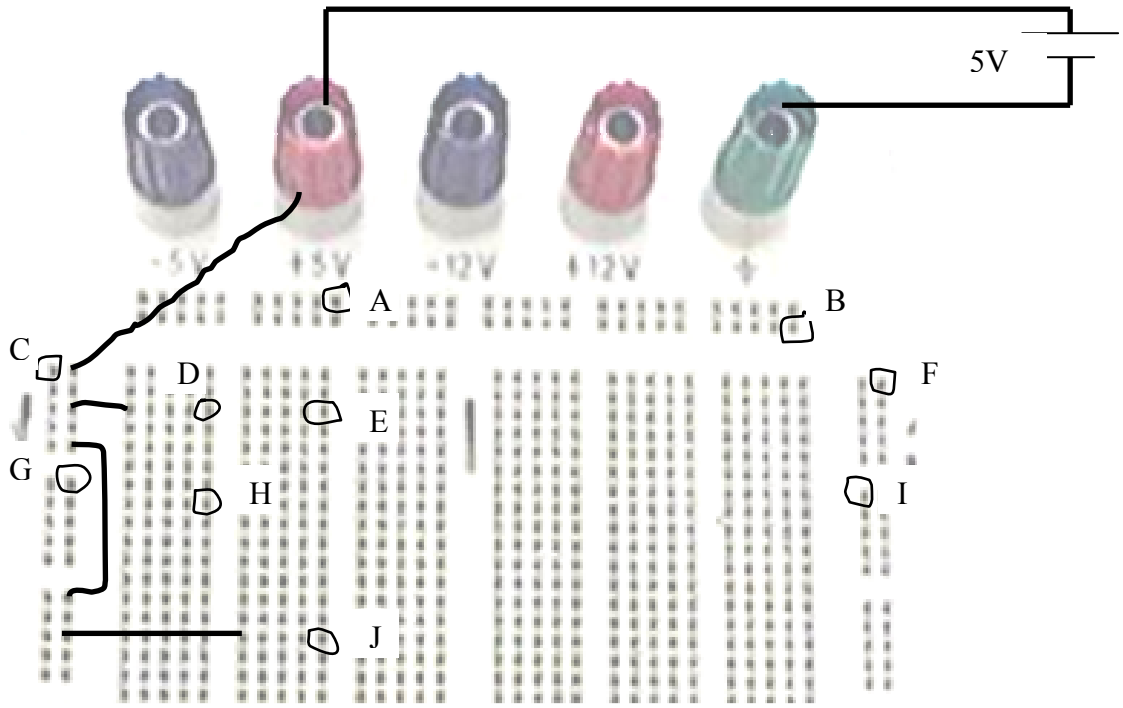


Perquè l'accés indirecte no està banquejat, no usa el registre BSR.

Per quin motiu els bits FSR1H<7:4> no s'usen?

Si tenim 4Kb d'espai d'adreçament, em calen 12 bits. No cal anar més enllà

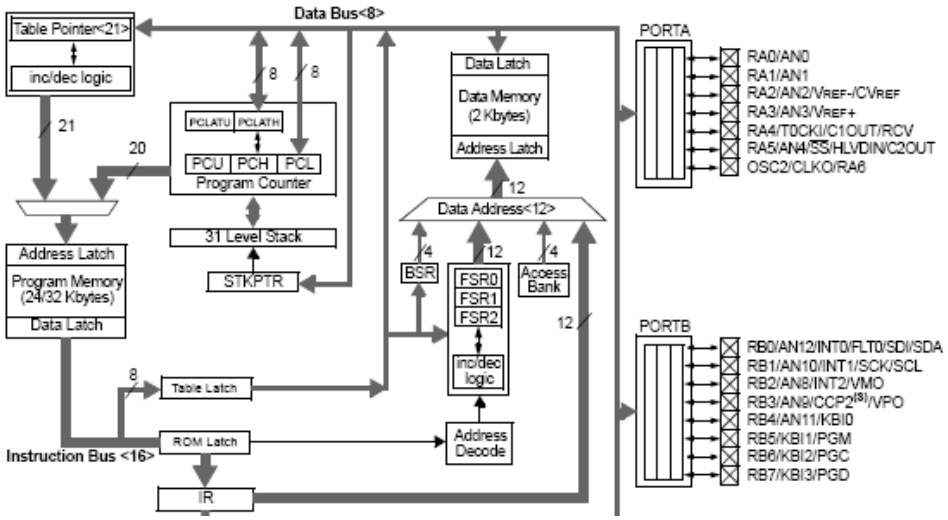
3) Considereu que les línies negres del següent dibuix són cables que fan connexió entre diferents parells de forats del proto-board.



Indiqueu les tensions que tindrem (respecte al càtode de la pila) als punts del circuit indicats:

	A	B	C	D	E	F	G	H	I	J
V	Aire	Aire	Aire	5V	Aire	Aire	5V	Aire	Aire	5V

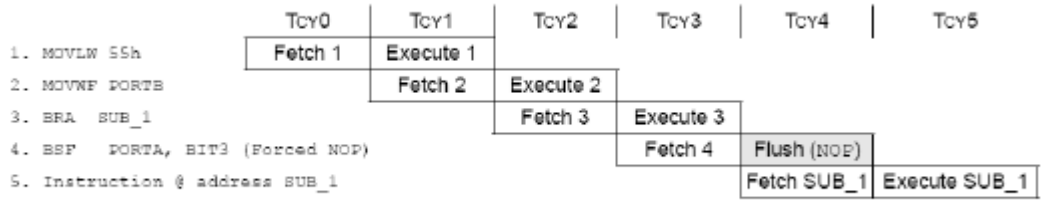
4) Quina és l'amplada en bits de la pila del 18F4550? Justifiqueu perquè calen aquest nombre de bits.



20 bits. Cal guardar les adreces de retorn, i aquestes són de 20 bits (1Mword d'espai d'adreçament)

Nom i Cognoms:_____ **Una possible solució**_____

5. Segons el codi que hi ha continuació indiqueu en el quadre següent quina acció realitzaria la CPU en cada cicle. El conjunt d'accions possibles a indicar serien: Fetch N (on N és el nº d'instrucció), Execute N o Flush. Suposeu que la CPU comença fent un fetch de la primera instrucció.



6. En executar la instrucció **MOVWF F,D,A** , quin valor hauria de tenir l'adreça **F** i el registre **BSR** si es vol escriure a l'adreça **EF1h** de la memòria RAM? Justifiqueu la vostra resposta.

BSR = 0Eh
F = F1h
L'adreçament a la RAM té 12 bits, els 4 bits més significatius es defineixen en els 4 bits menys significatius del registre BSR. Els altres 8 bits es defineixen en el camp F de la instrucció.

7. A partir de l'esquema de l'arquitectura de la pàgina anterior, és possible copiar un nombre imparell de bytes de la ROM de programa a la memòria RAM? Justifiqueu la vostra resposta.

Es pot observar que el "Table Pointer" té 21 bits i que aquest es poden utilitzar per adreçar la ROM. Com que es tracta d'una adreça completa, 21 bits en lloc dels 20 bits, es poden adreçar posicions de memòria imparells.

8. Quina són les principals diferències entre la instrucció GOTO i la instrucció BRA?

La instrucció GOTO és una instrucció de salt absolut i ocupa dos words on s'especifica l'adreça a on s'ha de saltar. La instrucció BRA és una instrucció de salt relatiu ocupant un word i s'especifica el valor que se li ha de sumar/restar al comptador de programa (PC). Amb el salt especificat en la instrucció BRA no es pot assolir tot l'espai de memòria degut al reduït nombre de bits reservats per especificar-lo.

TABLE 26-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
BIT-ORIENTED OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, d, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	n, s	Call Subroutine 1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
GOTO	n	Go to Address 1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	4
POP	—	Pop Top of Return Stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	—	Push Top of Return Stack (TOS)	1	0000	0000	0000	0101	None	
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	
RESET		Software Device Reset	1	0000	0000	1111	1111	All	
RETFIE	s	Return from Interrupt Enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with Literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	

TABLE 26-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
LITERAL OPERATIONS									
ADDLW	k	Add Literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND Literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR Literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Move Literal (12-bit) 2nd word	2	1110	1110	00ff	kkkk	None	
		to FSR(f) 1st word		1111	0000	kkkk	kkkk		
MOVLB	k	Move Literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move Literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply Literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with Literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from Literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR Literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS									
TBLRD*		Table Read	2	0000	0000	0000	1000	None	
TBLRD*+		Table Read with Post-Increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with Post-Decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with Pre-Increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2	0000	0000	0000	1100	None	
TBLWT*+		Table Write with Post-Increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with Post-Decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with Pre-Increment		0000	0000	0000	1111	None	