CC-CM-F - LIF6 Architecture matérielle et logicielle

Aucun document autorisé, durée 45 min, barème donné à titre indicatif. **N'oubliez pas de rendre le sujet complété, en indiquant bien votre numéro d'intercalaire**.

I Questions de cours (16 points)

Q.I.1) - (2 pts) Proposez une définition de ce qu'est le chemin de données d'une unité centrale de traitement.

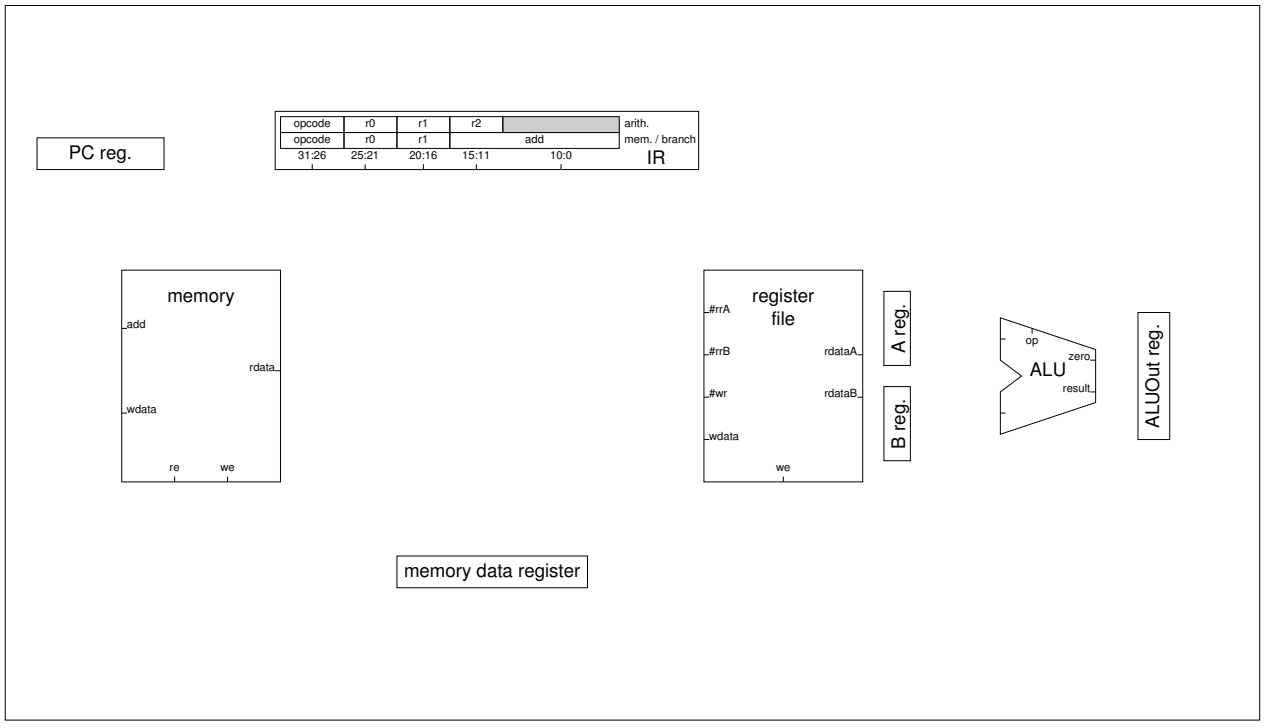
Q.I.2) - (3 pts) Dans le cours *ébauche d'un processeur*, nous nous sommes donnés les composants indiqués sur le schéma ci-dessous pour la conception du chemin de données. Il vous est demandé de complétez ce schéma en indiquant les multiplexeurs et les câblages nécessaires pour une instruction arithmétique :

$$\text{Reg[IR.r2]} \leftarrow \text{Reg[IR.r0]} \text{ OP } \text{Reg[IR.r1]}$$

Lors du cycle d'une instruction arithmétique, les actions à effectuer sont :

IR	←	Mem[PC]	(chargement de l'instruction dans IR)
PC	←	PC+4	(mise à jour du PC)
A	←	Reg[IR.r0]	(chargement du registre A)
B	←	Reg[IR.r1]	(chargement du registre B)
ALUOut	←	A OP B	(résultat de l'opération placé dans ALUOut)
Reg[IR.r2]	←	ALUOut	(stockage du résultat dans le banc de registres)

Ne complétez que le chemin de données, n'indiquez pas les signaux de contrôle.



Q.I.3) - (3 pts) Après avoir rappelé brièvement comment sont codées les instructions, définissez ce qu'est le *mode d'adressage* des opérandes d'une instructions. Donnez trois exemples de modes d'adressages, en illustrant chacun par une instruction en langage d'assemblage.

Q.I.4) - (2 pts) Comment se caractérise un jeu d'instruction CISC (*Complex Instruction Set Computer*) ?

Q.I.5) - (3 pts) Résumez la manière dont un assembleur (comme l'assembleur du LC-3) peut s'y prendre pour traduire un programme source en langage d'assemblage vers un code objet en langage machine.

Q.I.6) - (3 pts) Complétez le programme suivant afin qu'il calcule la longueur de la chaine de de caractères se trouvant à l'adresse `string`. Vous utiliserez `R0` comme pointeur pour parcourir la chaine de caractères, `R1` comme compteur de caractères. Le résultat sera stocké à l'adresse désignée par `res`.

```
        .ORIG x3000
        LEA R0,string   ; Chargement dans R0 de l'adresse de la chaine
        AND R1,R1,0     ; Mise a 0 du compteur
```

```
        HALT
; Chaine constante
string: .STRINGZ "Hello World"
res:    .BLKW #1
        .END
```

II Questions à réponses brèves (4 pts)

Q.II.1) - Lors de la conception d'une micro-architecture, qu'est ce qu'une méthodologie de synchronisation ?

Q.II.2) - Dans une certaine architecture, les instructions arithmétiques ne prennent aucune adresse : comment ces instructions peuvent être utilisées ?

Q.II.3) - Un banc de 16 registres 16 bits comporte 2 ports de lecture et un port d'écriture : combien de fils dédiés à l'identification des numéros de registre comporte-t'il ?

Q.II.4) - Une unité arithmétique et logique (UAL) prend deux mots de 16 bits en entrée, et fournit un résultat sur 16 bits. Elle est capable d'effectuer 7 opérations (ADD, SUB, MUL, DIV, NOT, AND, OR) distinctes. Combien de fils de contrôle cette UAL doit-elle prendre en entrée ?

Q.II.5) - Quelles sont les trois principales classes d'instructions d'un jeu d'instruction RISC, comme le LC-3 ?

Q.II.6) - Entre une architecture CISC et une RISC, laquelle présente le plus de registres architecturaux ?

Q.II.7) - Qu'est-ce qu'un label dans un langage d'assemblage ?

Q.II.8) - Sur le LC-3, quelque est l'interêt de l'instruction JMP, alors que l'on dispose déjà de BR ?

Récapitulatif des instructions du LC-3 utilisables ici :

syntaxe	**action**	**nzp**	**codage**															
			opcode				arguments											
			F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
NOT DR,SR	DR <- not SR	*	1 0 0 1				DR			SR			1 1 1 1 1 1					
ADD DR,SR1,SR2	DR <- SR1 + SR2	*	0 0 0 1				DR			SR1			0	0 0		SR2		
ADD DR,SR1,Imm5	DR <- SR1 + SEXT(Imm5)	*	0 0 0 1				DR			SR1			1	Imm5				
AND DR,SR1,SR2	DR <- SR1 and SR2	*	0 1 0 1				DR			SR1			0	0 0		SR2		
AND DR,SR1,Imm5	DR <- SR1 and SEXT(Imm5)	*	0 1 0 1				DR			SR1			1	Imm5				
LEA DR,label	DR <- PC + SEXT(PCoffset9)	*	1 1 1 0				DR			PCoffset9								
LD DR,label	DR <- mem[PC + SEXT(PCoffset9)]	*	0 0 1 0				DR			PCoffset9								
ST SR,label	mem[PC + SEXT(PCoffset9)] <- SR		0 0 1 1				SR			PCoffset9								
LDR DR,BaseR,Offset6	DR <- mem[BaseR + SEXT(Offset6)]	*	0 1 1 0				DR			BaseR			Offset6					
STR SR,BaseR,Offset6	mem[BaseR + SEXT(Offset6)] <- SR		0 1 1 1				SR			BaseR			Offset6					
BR[n][z][p] label	Si (cond) PC <- PC + SEXT(PCoffset9)		0 0 0 0				n	z	p	PCoffset9								
JMP BaseR	PC <- BaseR		1 1 0 0				0 0 0			BaseR			0 0 0 0 0 0					