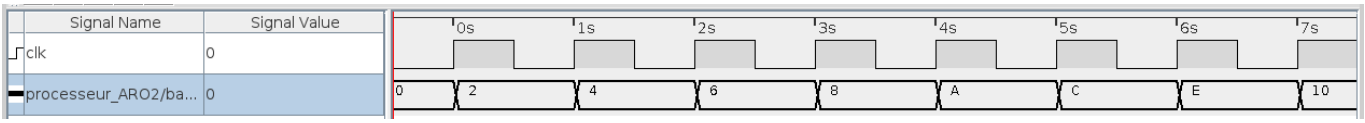
Labo 2 : Decode and Execute Piemontesi Trueb

Question 1

> Créez un petit programme de 5-6 instructions en assembleur. Buildez et chargez votre programme dans la mémoire d' instruction. Faites le chronogramme de l'execution de votre programme et vérifiez que le registre PC s'incrémente correctement.

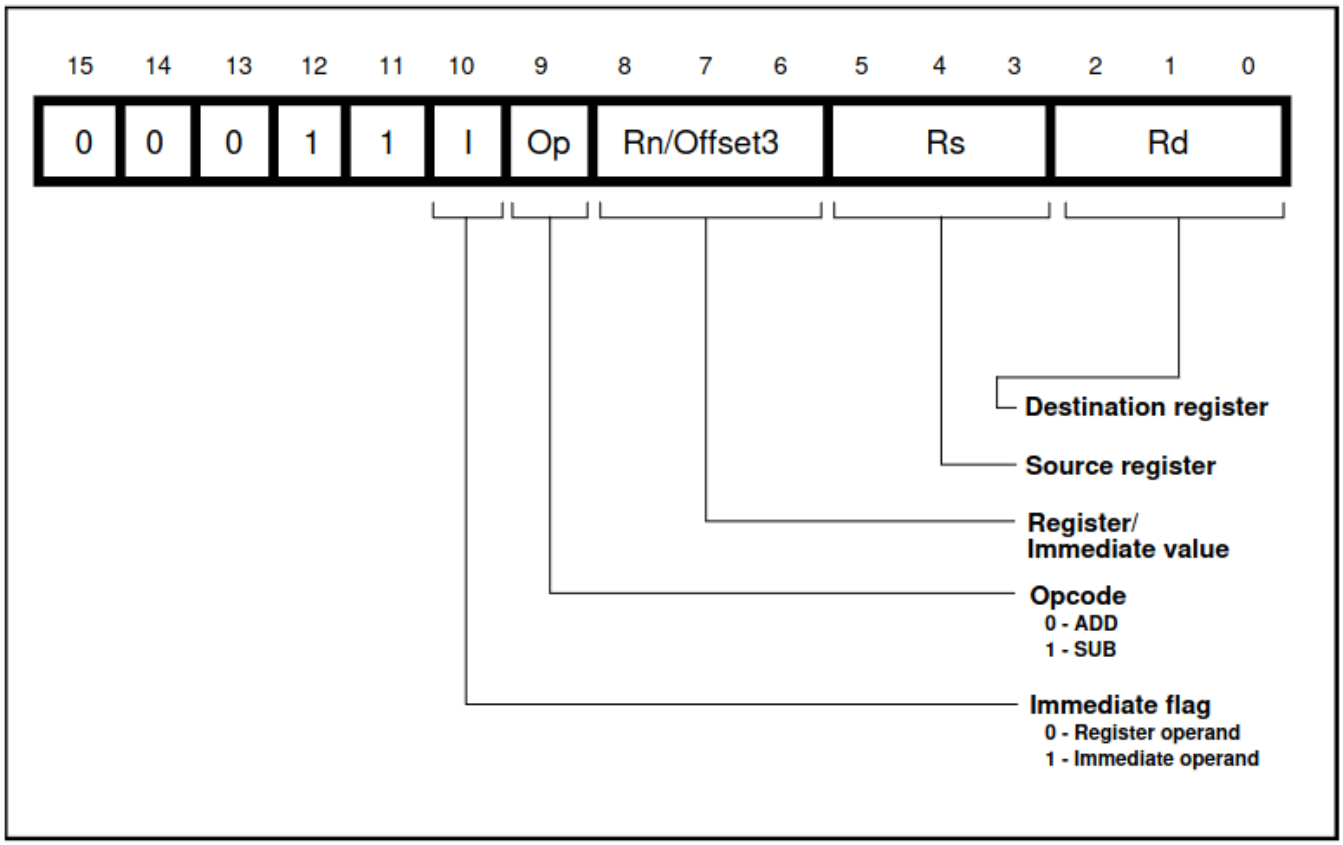


On peut voirt quer le PC s'incrémente correctement.

Question 2

> Lors d'une opération de type "Addition/substraction (add/substract)", sur quelles sorties de decode_isntr_splitter pourriez vous lire les informations indiquant les registres et offsets utilisés ?

5.2 Format 2: add/subtract

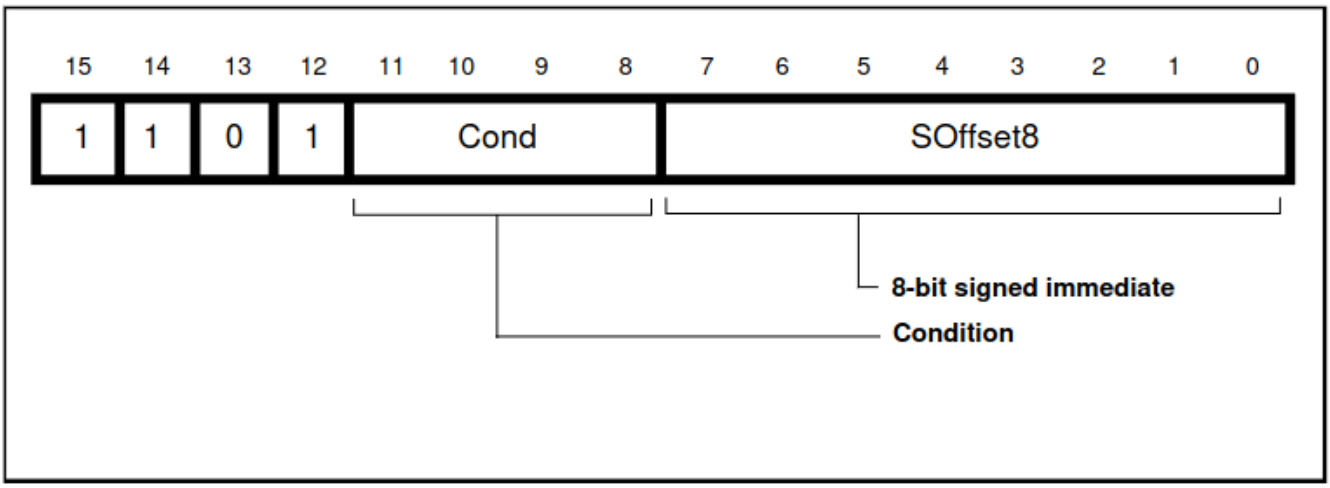


En suivant ce schéma on peut voir que le registre de destination est indiqué par les bits allant de **0** à **2** inclu, **(rd_2_0_o)**
le registre source est lui indiqué par les bits allant de **3** à **5** inclu **(rs_5_3_o)**
et que l'offset se trouve sur les bit allant de **6** à **8** inclu **(offset3_8_6_o ou rn_8_6_o)**.

Question 3

> Lors d'une opération de type "Branchement/saut conditionnel (conditional branch)", sur quelles sorties de decode_isntr_splitter pourriez vous lire les informations indiquant les bits de condition et l'offset utilisés ?

5.16 Format 16: conditional branch



En suivant ce schéma on peut voir que les bits de conditions sont ceux allant de **8** à **11** inclu **(cond_11_8_o)** et que l'offset se trouve sur les bit allant de **0** à **7** inclu **(offset8_7_0_o)**.

Question 4

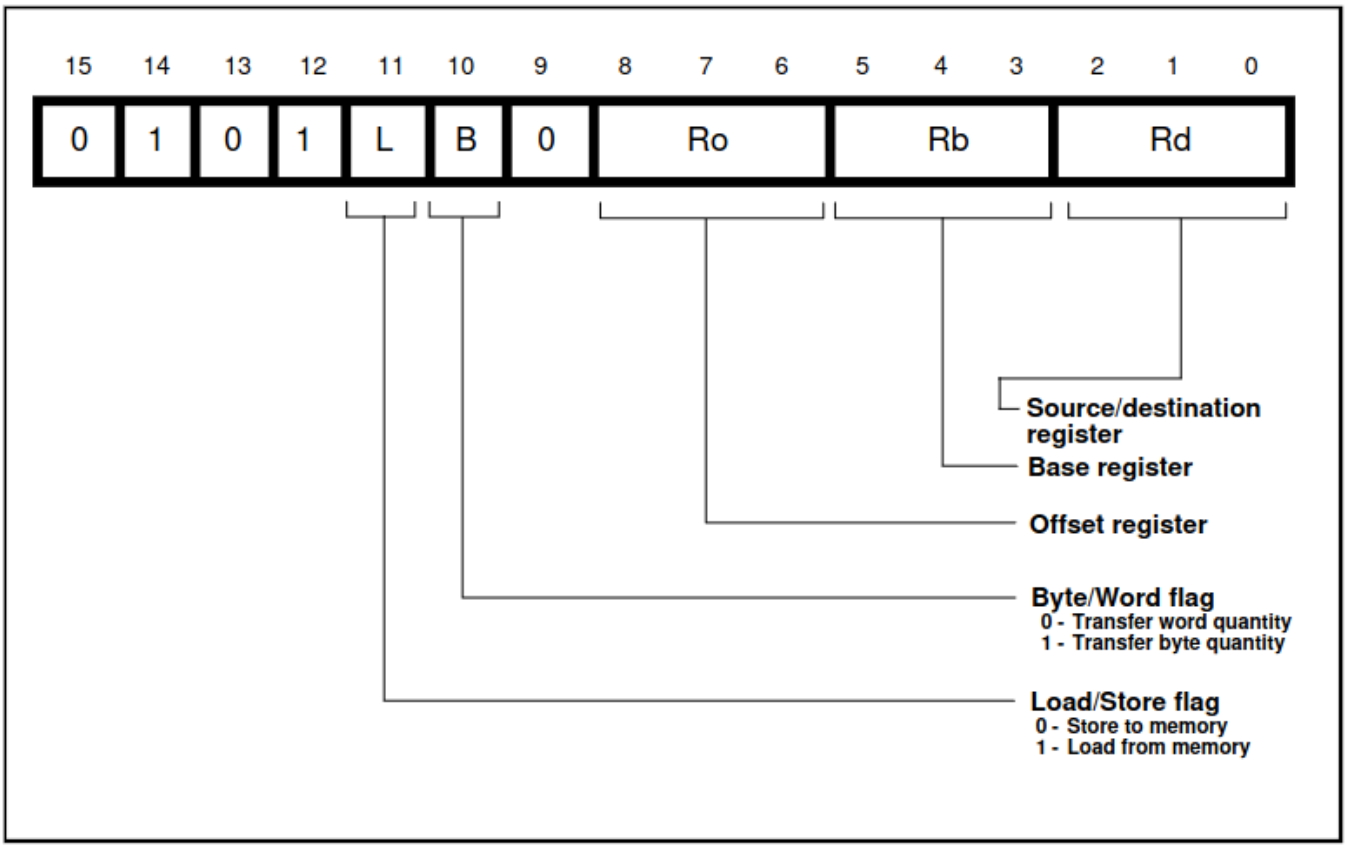
> Pour les 4 instructions travaillant uniquement avec des registres, quel est le nombre de bits de l'opcode ? Et pour les 4 instructions avec un offset immédiat ?

Pour les 4 instructions travaillant uniquement avec des registres :

Le nombre de bits de l'opcode est de **7 (bits allant de 9 à 15 inclu)**.

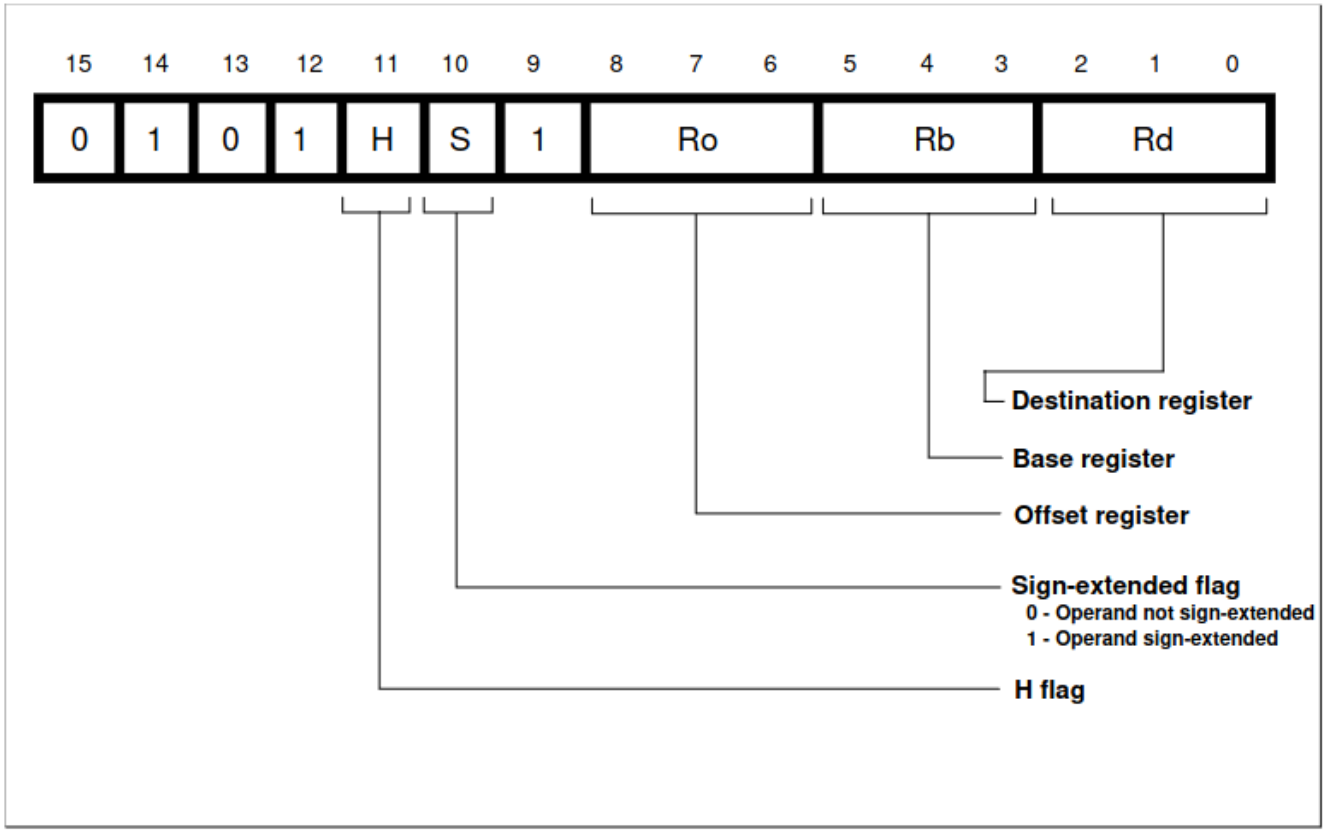
strb_r_r_r et ldrb_r_r_r :

5.7 Format 7: load/store with register offset



Pour **strh_r_r_r et ldrh_r_r_r** :

5.8 Format 8: load/store sign-extended byte/halfword

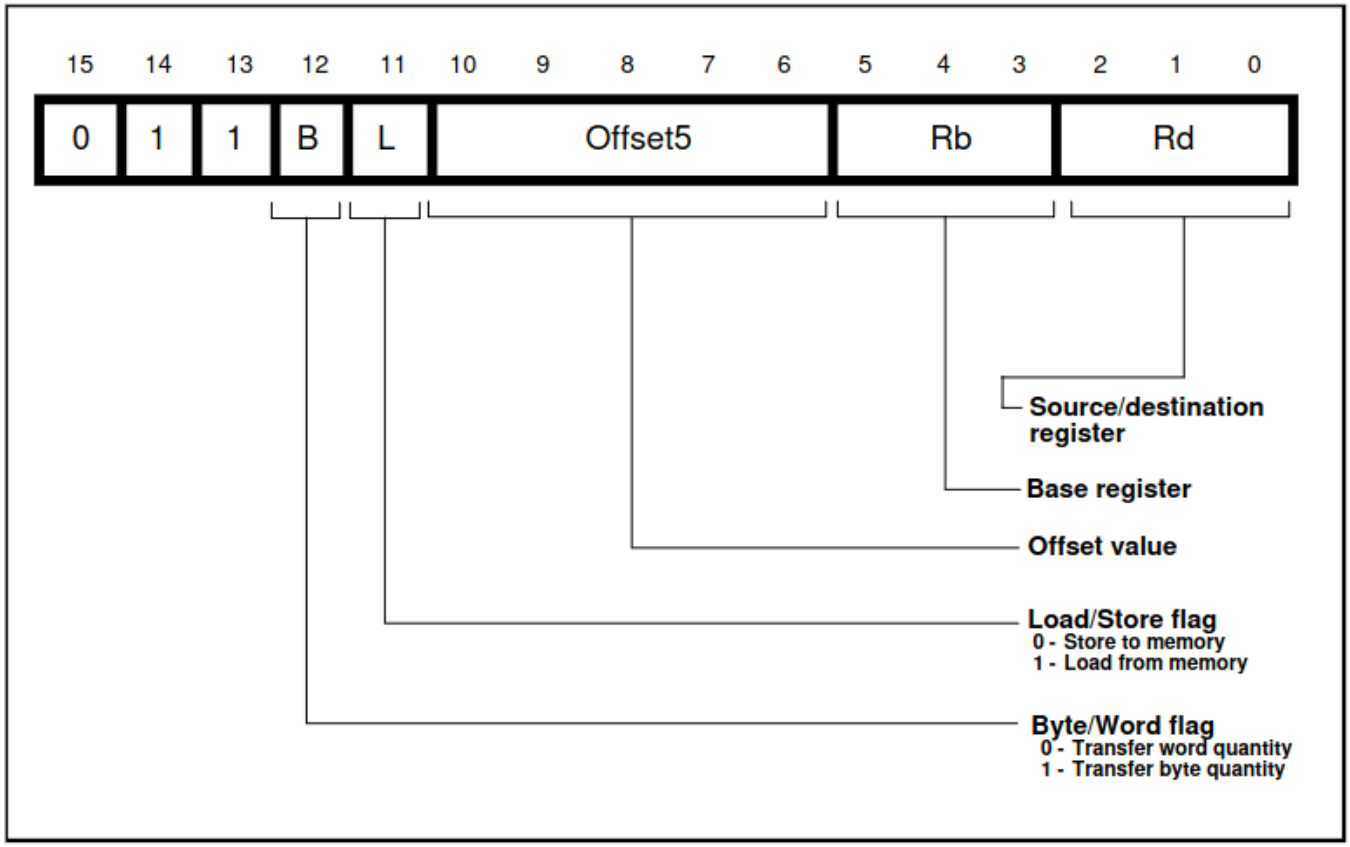


Pour les 4 instructions avec un offset immédiat :

Le nombre de bits de l'opcode est de **5 (bits allant de 11 à 15 inclu)**.

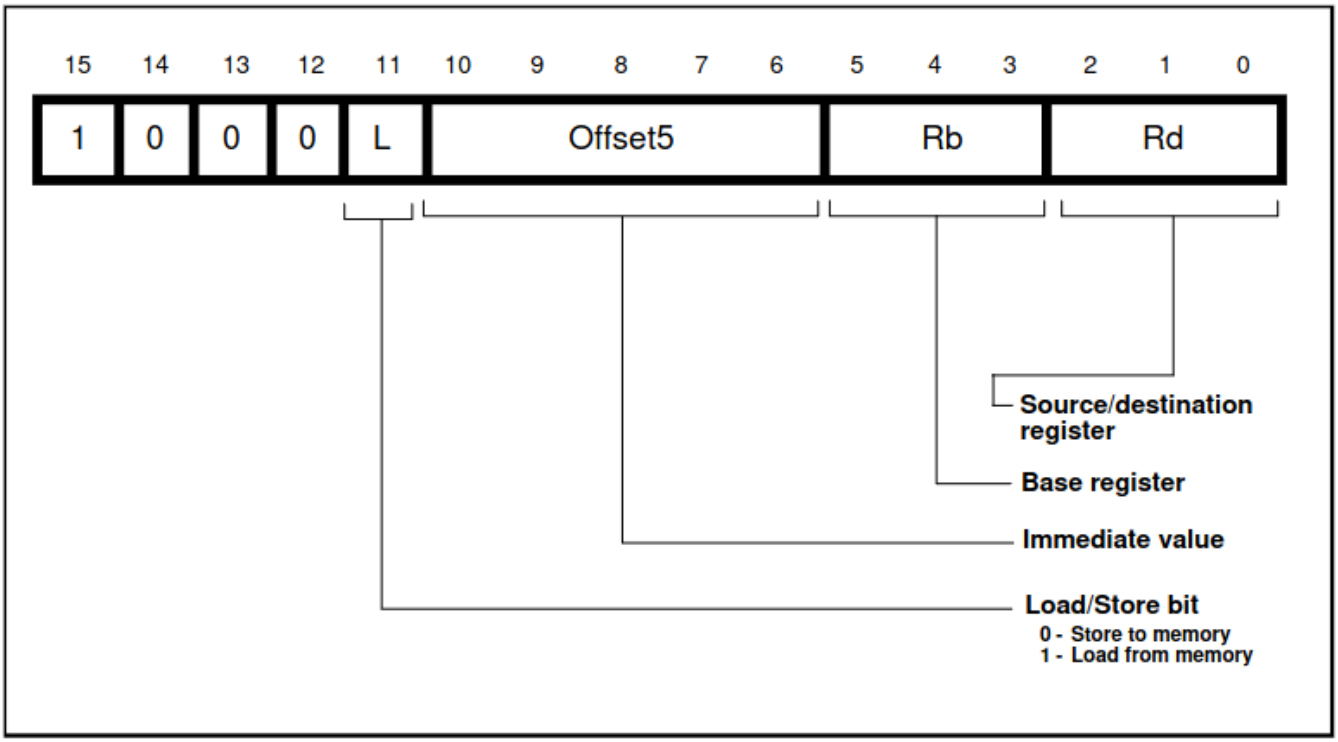
Pour **strb_r_r_imm et ldrb_r_r_imm**

5.9 Format 9: load/store with immediate offset



Pour **strh_r_r_imm et ldrh_r_r_imm** :

5.10 Format 10: load/store halfword



Eatpe 5.2

> Phase de test du circuit.

Etape	fetch_control_bus_o	decode_control_bus_o	reg_bank_control_bus_o	execute_control_bus_o	mem_control_bus_o	opcode_supported_unit
STEP_1	0x0024	0x0011	0x0002	0x04d0	0x0000	sub_r_imm_o
STEP_3	0x0020	0x0000	0x0002	0x2088	0x0005	ldrb_r_r_imm_s
STEP_4	0x0030	0x0000	0x0002	0x2008	0x000d	ldrb_r_r_r_s
STEP_7	0x0000	0x0000	0x0002	0x0401	0x0000	asr_r_t_imm_s
STEP_8	0x0003	0x0002	0x0000	0x1088	0x0000	b_cond_s
STEP_10	0x0001	0x0002	0x0000	0x0888	0x0000	b_incond_s

Etape	R0	R1	R2	R3	R4	SP	LR	PC
STEP_1	0x0020	0x001a	0x0026	0xfff4	0x0018	0x0000	0x0000	0x0010
STEP_3	0x0020	0x0024	0x00f4	0xfff4	0x0018	0x0000	0x0000	0x001c
STEP_4	0x0000	0x0024	0x00f4	0xfff4	0x0018	0x0000	0x0000	0x0020
STEP_7	0x0002	0x0024	0x000b	0xfff4	0x001c	0x0000	0x0000	0x003a
STEP_8	0x0000	0x0000	0x0000	0xfff4	0x001c	0x0000	0x0000	0x0042
STEP_10	0x0000	0x0000	0x0000	0xfff4	0x001c	0x0000	0x0000	0x0046