Les visiteurs
P-ARM

Thomas Prévost
Tom Niget
Emmeline Vouriot
Jinjin Wang

Sommaire

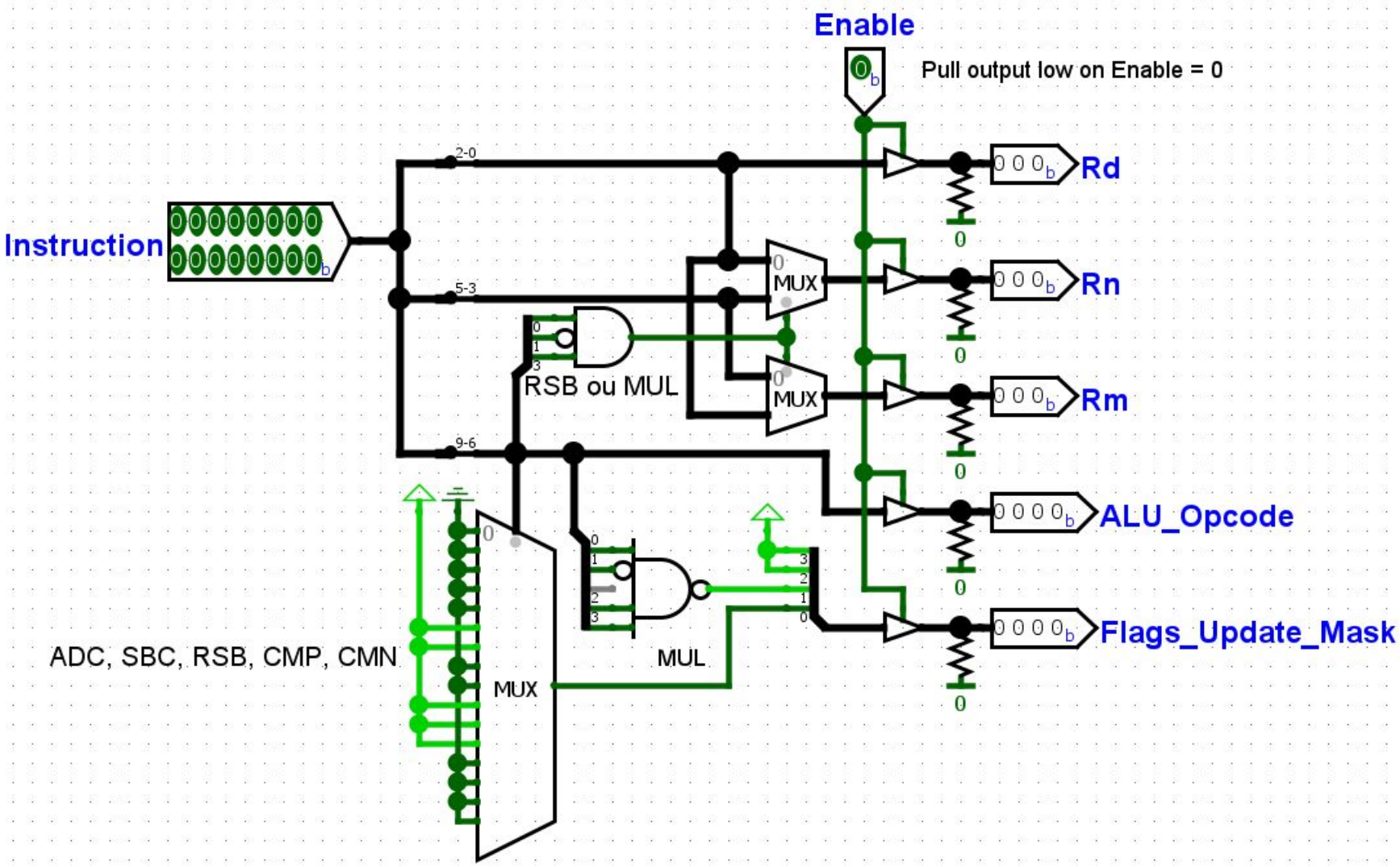
- Réaliser les circuits
- Assembleur
- Tests
- Pour aller plus loin

Sommaire

- Réaliser les circuits
- Assembleur
- Tests
- Pour aller plus loin

010000 <op> <reg1> <reg0>

OP	Instruction	rn	rd	rm	N	Z	C	V
0000	and	0	0	1	x	x	x	
0001	eor	0	0	1	x	x	x	
0010	lsl	0	0	1	x	x	x	
0011	lsr	0	0	1	x	x	x	
0100	asr	0	0	1	x	x	x	
0101	adc	0	0	1	x	x	x	x
0110	sbc	0	0	1	x	x	x	x
0111	ror	0	0	1	x	x	x	
1000	tst	0	0	1	x	x	x	
1001	rsb	1	0	0	x	x	x	x
1010	cmp	0	0	1	x	x	x	x
1011	cmn	0	0	1	x	x	x	x
1100	orr	0	0	1	x	x	x	
1101	mul	1	0	0	x	x		
1110	bic	0	0	1	x	x	x	
1111	mvn	0	0	1	x	x	x	


Enable
Pull output low on Enable = 0
Instruction
2-0
5-3
9-6
MUX
MUX
RSB ou MUL
Rd
Rn
Rm
ALU_Opcode
Flags_Update_Mask
ADC, SBC, RSB, CMP, CMN
MUX
MUL

Sommaire

- Réaliser les circuits
- Assembleur
- Tests
- Pour aller plus loin

Assembleur

- Utilisation des champs de bits du C

Définition

```c
//LSLS, LSRS, ASRS
typedef struct instruction_LLA
{
  unsigned int rd    :3;
  unsigned int rm    :3;
  unsigned int imm5  :5;
  unsigned int opcode:3;
  unsigned int idcode:2;
} Inst_shift;

//ADDS, SUBS, ADDS, SUBS
typedef struct
{
  unsigned int rd    :3;
  unsigned int rn    :3;
  unsigned int rm_imm  :3;
  unsigned int opcode:5;
  unsigned int idcode:2;
} Inst_ASA;
```

```c
//LSLS_imm, LSRS_imm, ASRS_imm
if ((num = sscanf(CmdLine,"%*s\tr%d, r%d, #%d",&rd,&rm,&imm5)) == 3){

  order_lla.idcode = 0;
  order_lla.opcode = inst;
  order_lla.rd = rd;
  order_lla.rm = rm;
  order_lla.imm5 = imm5;
  addr++;

  fprintf(out,"%04hx ",order_lla);

}

// LSLS, LSRS, ASRS
else if( num == 2 ){
  order_data.idcode = 16;
  order_data.opcode = inst + 2;  // lsls_imm == 0, lsls == 2
  order_data.rdn = rd;
  order_data.rm  = rm;
  addr++;
  fprintf(out,"%04hx ",order_data);
}
break;
```

Affectation

Output

Assembleur

- La sortie de l’assembleur

```
v2.0 raw

        sub     sp, #24         :b098
        movs    r0, #0          :2000
        str     r0, [sp, #20]   :9014
        str     r0, [sp, #16]   :9010
        str     r0, [sp, #12]   :900c
        movs    r0, #1          :2001
        str     r0, [sp, #8]    :9008
        movs    r0, #2          :2002
        str     r0, [sp, #4]    :9004
        movs    r0, #3          :2003
        str     r0, [sp]        :9000
        b       .LBB0_1         :defe
        ldr     r0, [sp, #32]   :9820
        ldr     r1, [sp, #12]   :990c
        cmp     r0, r1          :4288
        bne     .LBB0_3         :d104
        b       .LBB0_2         :defe
```

```
        ldr     r0, [sp, #24]   :9818
        ldr     r1, [sp, #28]   :991c
        adds    r0, r0, r1      :1840
        str     r0, [sp, #40]   :9028
        b       .LBB0_3         :defe
        ldr     r0, [sp, #32]   :9820
        ldr     r1, [sp, #8]    :9908
        cmp     r0, r1          :4288
        bne     .LBB0_5         :d104
        b       .LBB0_4         :defe
        ldr     r0, [sp, #24]   :9818
        ldr     r1, [sp, #28]   :991c
        subs    r0, r0, r1      :1a40
        str     r0, [sp, #40]   :9028
        b       .LBB0_5         :defe
        ldr     r0, [sp, #32]   :9820
        ldr     r1, [sp, #4]    :9904
        cmp     r0, r1          :4288
        bne     .LBB0_7         :d104
        b       .LBB0_6         :defe
```

Sommaire

- Réaliser les circuits
- Assembleur
- Tests
- Pour aller plus loin

Tests

Comment générer des tests exhaustifs ? Nous avons programmé un générateur de tests aléatoires en C++:

```cpp
#include <iostream>
#include "tester.h"

using namespace std;

int main()
{
    Tester tester("C:\\Users\\thoma\\Documents\\travail\\polytech\\SI3\\archi\\tests\\test_controller.txt", 50);
    tester.generateTest(Tester::LDR);
    return 0;
}
```

Tests

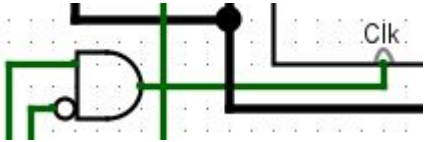
Pour cela nous nous sommes basés sur la documentation, ce qui a permis en outre de mieux comprendre le fonctionnement du projet.

Sommaire

- Réaliser les circuits
- Assembleur
- Tests
- Pour aller plus loin

Problèmes rencontrés pendant la création

- Modification de registres par l'instruction STR
  → Ajout d'une vérification de !Store pour l'horloge du banc de registres

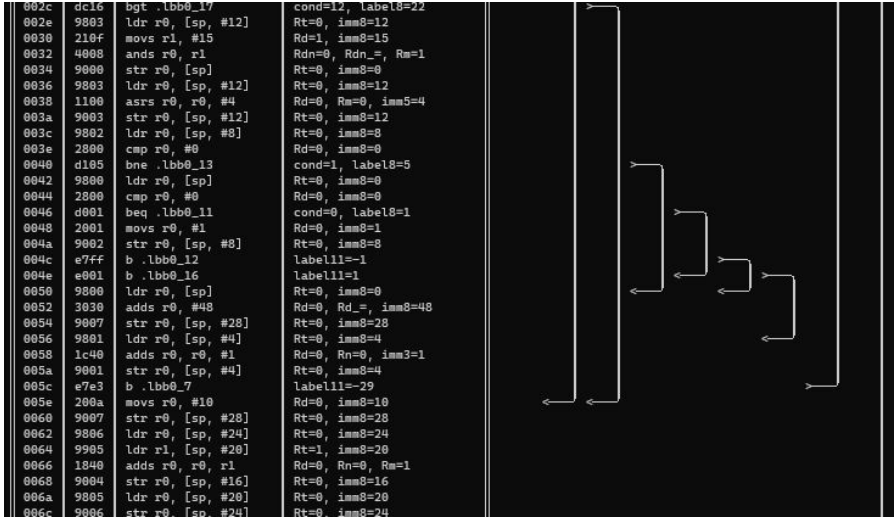


- Problèmes de simultanéité quand l'instruction change alors qu'elle est en train d'être décodée
  → Passage de PC et de la bascule PC_Hold sur le front descendant pour empêcher les conflits
- Problèmes liés à la documentation, par ex. erreurs d'ordre d'opérandes
  → Signalement des erreurs aux enseignants

Pour aller plus loin

- Création d'un nouvel assembleur entièrement dynamique pour permettre l'ajout facile de nouvelles instructions (pour supporter des programmes plus complexes)
- Visualisation graphique du flux d'exécution du programme
- Passage de PC sur 16 bits
- Ajout d'instructions
  - Branche avec décalage sur 11 bits
  - Add, sub, cmp avec immédiat 8 bits

```
# 01 - move shifted register
"lsls {Rd}, {Rm}, {imm5}": (0b000_00, "imm5", "Rm", "Rd"),
"lsrs {Rd}, {Rm}, {imm5}": (0b000_01, "imm5", "Rm", "Rd"),
"asrs {Rd}, {Rm}, {imm5}": (0b000_10, "imm5", "Rm", "Rd"),
"movs? {Rd}, {Rm}": "lsls {Rd}, {Rm}, #0",
# 02 - add/subtract
"adds {Rd}, {Rn}, {Rm}": (0b000_11_00, "Rm", "Rn", "Rd"),
"subs {Rd}, {Rn}, {Rm}": (0b000_11_01, "Rm", "Rn", "Rd"),
"adds {Rd}, {Rn}, {imm3}": (0b000_11_10, "imm3", "Rn", "Rd"),
"subs {Rd}, {Rn}, {imm3}": (0b000_11_11, "imm3", "Rn", "Rd"),
# 03 - move/compare/add/subtract immediate
"movs {Rd}, {imm8}": (0b001_00, "Rd", "imm8"),
"cmp {Rd}, {imm8}": (0b001_01, "Rd", "imm8"),
"adds {Rd}, ({Rd_}, )?{imm8}": (0b001_10, "Rd", "imm8"),
"subs {Rd}, ({Rd_}, )?{imm8}": (0b001_11, "Rd", "imm8"),
# 04 - ALU operations
"ands {Rdn}, ({Rdn_}, )?{Rm}": (0b010000_0000, "Rm", "Rdn"),
"eors {Rdn}, ({Rdn_}, )?{Rm}": (0b010000_0001, "Rm", "Rdn"),
"lsls {Rdn}, ({Rdn_}, )?{Rm}": (0b010000_0010, "Rm", "Rdn"),
"lsrs {Rdn}, ({Rdn_}, )?{Rm}": (0b010000_0011, "Rm", "Rdn"),
"asrs {Rdn}, ({Rdn_}, )?{Rm}": (0b010000_0100, "Rm", "Rdn"),
```



Pour aller plus loin

Ajout de périphériques en MMIO facilement utilisables depuis le code

```
choice = READKEY();

if (choice == '+')
    RES = a + b;
else if (choice == '-')
    RES = a - b;
else if (choice == '*')
```

```
while (1)
{
    PIXSET(tx, ty);
    PIXSET(x, y);

    SCRUPD();
```

```
    PUTCHAR('\n');
    PUTCHAR('R');
    PUTCHAR('=');
    PRINTRES_SIGN();

    while (!KEYBeof);
```

```
while(1)
{
    NOTE(76);
    NOTE(75);
    NOTE(76);
    NOTE(75);
    NOTE(76);
```