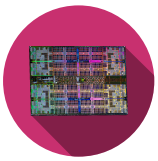


Tous les exercices CAr

Exercices Architecture des ordinateurs

1 | Puce & silicon fabrication

1.1 Fabrication

Nous souhaitons produire une puce dont la taille est de 15mm x 20mm. Les puces seront fabriquées sur des plaquettes de silicium de 30cm de diamètre. Nous achetons des lingots pour 20'000CHF, chacun d'eux pouvant être découpé en 100 plaquettes. Nous utilisons un procédé qui permet d'obtenir une moyenne de $0,12$ de défauts par cm^2.

a) Quel est le rendement de fabrication ?
b) Combien de puces peuvent tenir sur une plaquette ?
c) Combien de bonnes puces peut-on obtenir par tranche ?
d) Quel est le coût d'un bon puces ?

fun/fabrication-01

1.2 Fabrication

Nous souhaitons produire une puce dont la taille est de $150mm^2$. Les puces seront fabriquées sur des plaquettes de silicium de 20cm de diamètre et d'une épaisseur de 5mm. Nous achetons des lingots pour 30'000CHF, chacun a une longueur de 60cm. Nous utilisons un procédé qui donne une moyenne de $0,2$ de défauts par cm^2, le rendement du boîtier est de 80% et le coût du boîtier est de 0,06CHF par puce.

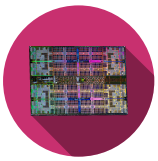
a) Combien de plaquettes par lingot ?
b) Quel est le coût par wafer ?
c) Quel est le coût par cm^2 de la wafer ?
d) Combien de puce peuvent tenir sur un wafer ?
e) Combien de bonnes puce sont-ils sur une tranche de silicium ?
f) Quel est le coût de la puce finale ?

fun/fabrication-02

1.3 Fabrication

Un procédé utilise une tranche de silicium (wafer) de 20cm de diamètre et une puce (die) de $52,3mm^2$. Le lingot coûte 6000CHF et peut être découpé en 30 tranches de silicium (wafers). Le procédé permet d'obtenir $0,03$ défaut par mm^2.

a) Combien coûte un wafer ?



b) Quel est le rendement (Yield) ?
c) Combien de die peuvent tenir sur une tranche de silicium (wafer) ?
d) Quel est le coût par puce (die) ?

fun/fabrication-03

2 | La loi de Moore & l'échelle de Denard

2.1 Échelle de Dennard

Si le nombre de transistors par unité de surface (c'est-à-dire sur la même taille de puce) double tous les 24 mois :

a) De quel facteur la longueur du côté d'un transistor, en supposant qu'il soit carré, diminue-t-elle chaque année ?
b) En partant d'une longueur de $18,4$ nm (2014), quelle sera la taille d'un transistor en 2025 ? Comment cela se compare-t-il à la valeur de $6,75 \mu m$?

fun/dennardscaling-01

2.2 La consommation dynamique d'un circuit CMOS est :

- ☐ directement proportionnel à la fréquence
- ☐ inversement proportionnel à la fréquence
- ☐ directement proportionnel au carré de la tension d'alimentation

fun/dennardscaling-02

3 | Consommation d'énergie

3.1 Autonomie de la batterie du téléphone portable

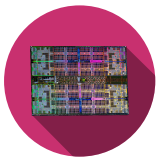
Un téléphone portable particulier possède une batterie de 8W-hr et fonctionne à 0,707V. Supposons que, lorsqu'il est utilisé, le téléphone portable fonctionne à 2GHz. La capacité totale du circuit est de 10 nF, et le facteur d'activité est de 0,05. Lorsque la voix ou les données sont actives (10 % de son temps d'utilisation), il émet également 3 W de puissance par son antenne. Lorsque le téléphone n'est pas utilisé, la puissance dynamique tombe à presque zéro car le traitement du signal est désactivé. Mais le téléphone consomme également 100mA de courant de repos, qu'il soit utilisé ou non.
Déterminer l'autonomie de la batterie du téléphone

a) s'il n'est pas utilisé
b) s'il est utilisé en permanence

fun/powerconsumption-01

1 | Benchmark du processeur & Performance

1.1 Les quelles des propositions suivantes sont correctes ?



- [ ] Le temps wall clock time est le temps total écoulé, y compris les E/S, les frais généraux du système d'exploitation, etc.
- [ ] Le multithreading améliore le débit d'un processus.
- [ ] Le temps CPU n'inclut pas le temps d'E/S.
- [ ] Le multithreading améliore le temps d'exécution d'un processus.

per/benchmark-01

1.2 Qu'est-ce que le débit (throughput)?

- [ ] performance par Watt (le nombre de FLOPS par Watt)
- [ ] le taux de traitement du travail (n tâches/seconde)
- [ ] le temps entre le début et la fin de l'événement/tâche/programme (n secondes)
- [ ] le pourcentage de temps pendant lequel un système est opérationnel

per/benchmark-02

1.3 Qu'est-ce que le SPEC?

- [ ] est une benchmark développée pour mesurer la performance basée sur la dernière fonctionnalité de l'application Java.
- [ ] est un benchmark qui évalue les caractéristiques de puissance et de performance des ordinateurs de classe serveur de volume.
- [ ] est la norme mondiale pour mesurer les performances graphiques sur la base d'applications professionnelles.
- [ ] est une combinaison de bechmark conçue pour fournir des mesures de performance qui peuvent être utilisées pour comparer des charges de travail informatiques intensives sur différents systèmes informatiques.

per/benchmark-03

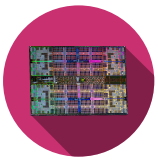
1.4 Quel est l'objectif du Benchmark EEMBC ?

- [ ] pour évaluer les performances des microprocesseurs embarqués
- [ ] pour évaluer les performances du calcul des nombres entiers
- [ ] pour mesurer l'efficacité énergétique de différents systèmes informatiques
- [ ] pour évaluer les performances en virgule flottante

per/benchmark-04

1.5 Lequel des éléments suivants est une mesure de l'efficacité énergétique ?

- [ ] flops
- [ ] MIPS
- [ ] Performance par watt
- [ ] Consommation électrique



per/benchmark-05

1.6 La consommation d'énergie et les performances par watt sont toutes deux importantes pour un système embarqué.

- ☐ Vrai
- ☐ Faux

per/benchmark-06

1.7 Performances du processeur

Un programme est composé de 5′000 instructions en virgule flottante et de 25′000 instructions en nombres entiers. Le processeur A a une fréquence d'horloge de $2,0$ GHz. Les instructions en virgule flottante prennent 7 cycles et les instructions en nombre entier prennent 1 cycle.

a) Combien de temps faut-il à ce processeur pour exécuter le programme ?
b) Quel est le CPI moyen de ce processeur pour le programme donné ?
c) Le processeur A exécute le programme 2 composé de 100′000 instructions en virgule flottante et de 50′000 instructions en virgule entière. Quel est le CPI moyen de ce programme ?
d) Le processeur B a un CPI moyen pour le programme 2 de $3,5$. Sa fréquence d'horloge est de $1,8$ GHz. Combien de temps faut-il pour exécuter le programme ?
e) Quel processeur est le plus rapide et de combien ?

Le processeur _____ est _____ fois plus rapide que le processeur _____.

per/performance-01

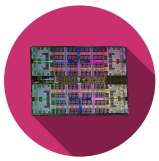
1.8 Performances du processeur

Considérons les deux conceptions de machines suivantes avec leurs CPI respectifs pour divers types d'instructions. L'ordinateur A et l'ordinateur B ont le même jeu d'instructions :

Instruction Type	CPI_A	CPI_B	Compiler 1 Mix
Data Manipulation	1.5	1.0	25%
Arithmetic	1.0	1.5	30%
Shifting	1.0	1.2	10%
Branching	4.0	2.0	25%
Multiply	20	12	10%

a) Quel est le CPI moyen pour chacun des ordinateurs utilisant ce programme ?
b) L'ordinateur A a un temps de cycle clock de $0,5$ns. L'ordinateur B tourne à $1,8$GHz. Rédigez une déclaration quantitative comparant les deux ordinateurs.
c) Quelle devrait être la fréquence d'horloge de l'ordinateur le plus lent pour obtenir des performances égales à celles de l'ordinateur le plus rapide ?

per/performance-02



1.9 Performances du processeur

Un CPU fonctionne à une fréquence de base de 2GHz. Il exécute un programme de 5 millions d'instructions avec la combinaison d'instructions donnée. newline Quel est le temps d'exécution du programme ?

Instruction	Frequency	CPI_{instr}
ALU	50%	3
Load	20%	5
Store	10%	4
Branch	20%	3

per/performance-03

1.10 Performances du processeur

Un CPU est conçue pour obtenir des performances optimales sur un programme donné qui présentant les caractéristiques suivantes. 25% de toutes les instructions sont des instructions en virgule flottante avec un CPI moyen de 4.0, de plus le programme contient 2% d'instructions FPSQR avec un CPI moyen de 20. Toutes les autres instructions ont un CPI moyen de 1.33.
Il existe deux alternatives de conception :

1. Diminuer le CPI des instructions FPSQR à 2.0
2. Diminue la moyenne du CPI de toutes les instructions en virgule flottante à 2.5

Quel est le meilleur choix ?

per/performance-04

1.11 Performances du processeur

Nous voulons acheter un nouvel ordinateur. Il exécutera principalement les programmes P_1 et P_2. Quel poids doivent avoir les programmes (w_1 et w_2) pour que :

a) CPU A soit le meilleur achat ?
b) CPU B soit le meilleur achat ?
c) CPU C soit le meilleur achat ?

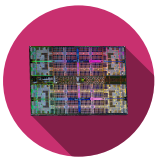
Program	CPU_A	CPU_B	CPU_C
Program P_1 (sec)	1	10	100
Program P_2 (sec)	100	10	1

per/performance-05

1.12 Performances du processeur

Étant donné les performances suivantes de deux programmes sur trois CPU, utilisez la moyenne géométrique pour calculer quel ordinateur est le plus rapide :

a) CPU A est le plus rapide !



b) CPU B est le plus rapide !
c) CPU C est le plus rapide !

Program	CPU_A	CPU_B	CPU_C
P_1 (sec)	40	15	20
P_2 (sec)	40	1000	150

per/performance-06

1.13 Performances du processeur

Calculez le CPI moyen pour 5 millions d'instructions des fréquences d'instruction suivantes :

Instruction	Frequency	CPI_{instr}
ALU	40%	4
Load	30%	6
Store	5%	5
Branch	25%	4

La fréquence d'horloge du processeur est de 2 GHz.

per/performance-07

1.14 Quelle est la meilleure mesure pour comparer les performances ?

- ☐ moyenne arithmétique
- ☐ moyenne géométrique
- ☐ médiane
- ☐ performance maximale
- ☐ moyenne harmonique

per/performance-08

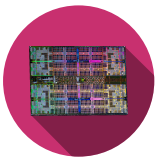
1.15 Performances du processeur

Calculez le temps d'exécution en ms, en supposant que l'on utilise un CPU avec les fréquences d'instruction suivantes :

Instruction	Frequency	CPI_{instr}
ALU	45%	5
Load	25%	6
Store	10%	5
Branch	20%	3

Pour 2 millions d'instructions et une fréquence CPU de 3GHz.

per/performance-09



1.16 Loi d'amdahl

Une amélioration de l'unité d'exécution en virgule flottante a généré des instructions en virgule flottante 2x plus rapides. En moyenne, 10% de toutes les instructions sont des instructions à virgule flottante pour ce processeur.

Quel sera le gain de vitesse global ?

per/amdahls-law-01

1.17 Loi d'amdahl

Nous voulons une accélération globale de 2 et pouvons accélérer les instructions en virgule flottante par 4 fois.

Quelle devrait être la fraction des instructions en virgule flottante ?

per/amdahls-law-02

1.18 Loi d'amdahl

Un programme se compose de deux éléments différents. L'élément A a une durée de 15 unités de temps et l'élément B une durée de 5 unités de temps. Il existe deux variantes d'optimisation :
1. Optimation de la partie A par deux fois
2. Optimation de la partie B par cinq fois

Quelle optimisation est la plus avantageuse ? Quelles sont les implications ?

per/amdahls-law-03

1 Implementation

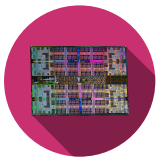
1.1 Quelle est la principale différence entre un système en temps réel dur et un système en temps réel souple ?

- ☐ Dans un système de temps réel dur, toutes les limites doivent être respectées, alors que dans un système de temps réel mou, certaines échéances peuvent être dépassées.
- ☐ Dans un système de temps réel souple, tous les délais doivent être respectés, alors que dans un système de temps réel rigide, certains délais peuvent parfois être dépassés.

imp/implementation-01

1.2 Qu'est-ce qu'un système embarqué ?

- ☐ Tout système informatique est un système embarqué
- ☐ Un système informatique avec une fonction dédiée, souvent avec des contraintes de temps, est un système embarqué.
- ☐ Un ordinateur à usage général avec moins de 1 Go de RAM est un système embarqué.



- [ ] Un système qui possède un processeur ARM est un système embarqué

imp/implementation-02

1.3 Un temps d'exécution plus rapide signifie moins d'énergie.

- [ ] Vrai
- [ ] Faux

imp/implementation-03

1.4 Pourquoi de plus en plus de SOC sont développés à la place des CPU ?

- [ ] restreindre la disponibilité de l'énergie
- [ ] accélérer les fonctions souvent utilisées
- [ ] les deux ci-dessus

imp/implementation-04

1 Instruction-Set Architecture

1.1 Code C simple vers assembleur RISC-V

Compilez le code C suivant en assembleur RISC-V.

a)
```
a = b + c;
```

b)
```
a = b + c - d;
```

c)
```
a = b + 6;
```

d)
```
// int is a 32-bit signed word
int a = -372;
int b = a + 6;
```

e)
```
int a = 0xFEDC8765;
```

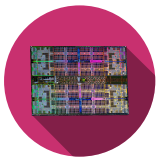
f)
```
int a = 0xFEDC8EAB;
```

isa/c-to-riscv-01

1.2 Code C algorithmique vers assembleur RISC-V

Compilez le code C suivant en assembleur RISC-V.

a)



```c
if (i == j){
  f = g + h;
}
f = f - i;
```

b)

```c
if (i == j){
  f = g + h;
}
else {
  f = f - i;
}
```

c)

```c
// add the numbers from 0 to 9
int sum = 0;
int i;

for (i=0; i!=10; i=i+1){
  sum = sum + i;
}
```

d)

```c
// add the powers of 2 from 1 to 100
int sum = 0;
int i;

for (i=1;i<101; i=i*2){
  sum = sum + i;
}
```

e)

```c
int array[5];
array[0] = array[0] * 2;
array[1] = array[1] * 2;
```

f)

```c
int array[1000];
int i;

for (i=0; i<100; i=i+1){
  array[i] = array[i] * 8;
}
```

g)

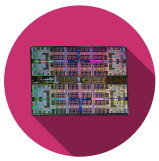
```c
char str[80] = "CAT";
int len = 0;

// compute length of string
while (str[len]) len++;
```

isa/c-to-riscv-02

1.3 Code machine vers assembleur RISC-V

Decoder le code machine suivant en assembleur RISC-V.

a) **`0x41FE 83B3`**



b) **`0xFDA4 8393`**

isa/machinecode-to-riscv-01

1.4 Opérations logiques sur registres

Exécutez le code assembleur et indiquez le contenu du registre cible **`rd`** si les registres sources **`rs`** contiennent les données suivantes :

```
s1 = 0x46A1 F1B7
s2 = 0xFFFF 0000
```

a)
```
and s3, s1, s2
```

b)
```
or s4, s1, s2
```

c)
```
xor s5, s1, s2
```

isa/riscv-execution-01

1.5 Opérations logiques sur valeurs

Exécutez le code assembleur et indiquez le contenu du registre cible **`rd`** si les registres sources **`rs`** contiennent les données suivantes :

```
t3 = 0x3A75 0D6F
```

a)
```
and s5, t3, -1484
```

b)
```
or s6, t3, -1484
```

c)
```
xor s7, t3, -1484
```

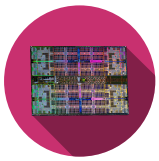
isa/riscv-execution-02

1.6 Multiplications en RISC-V

Exécutez le code assembleur et indiquez le contenu du registre cible **`rd`** si les registres sources **`rs`** contiennent les données suivantes :

```
s1 = 0x4000 0000
s2 = 0x8000 0000
```

```
mulh s4, s1, s2
mul  s3, s1, s2
```



isa/riscv-execution-03

1.7 Division et modulo

Exécutez le code assembleur et indiquez le contenu du registre cible **`rd`** si les registres sources **`rs`** contiennent les données suivantes :

```
s1 = 0x0000 0011
s2 = 0x0000 0003
```

```
div s3, s1, s2
rem s4, s1, s2
```

isa/riscv-execution-04

1.8 Type R vers code machine

Encoder l'assembleur RISC-V suivant en code machine.

a)
```
add s2, s3, s4
```

b)
```
sub t0, t1, t2
```

c)
```
sll s7, t0, s1
```

d)
```
xor s8, s9, s10
```

e)
```
srai t1, t2, 29
```

isa/riscv-to-machinecode-01

1.9 Type I vers code machine

Encoder l'assembleur RISC-V suivant en code machine.

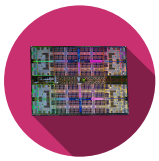
a)
```
addi s0, s1, 12
```

b)
```
addi s2, t1, -14
```

c)
```
lw t2, -6(s3)
```

d)
```
lh s1, 27(zero)
```

e)
```
lb s4, 0x1F(s4)
```



isa/riscv-to-machinecode-02

1.10 Type S vers code machine

Encoder l'assembleur RISC-V suivant en code machine.

a)
```
sw t2, -6(s3)
```

b)
```
sh s4, 23(t0)
```

c)
```
sb t5, 0x2D(zero)
```

isa/riscv-to-machinecode-03

1.11 Système temps réel

Quelle est la principale différence entre un système en temps réel « dur » et un système en temps réel « souple » ?

- ☐ Dans un système à temps réel « hard », tous les délais doivent être respectés, alors que dans un système à temps réel « soft », certains délais peuvent occasionnellement ne pas être respectés.
- ☐ Dans un système à temps réel « soft », tous les délais doivent être respectés, alors que dans un système à temps réel « hard », certains délais peuvent occasionnellement ne pas être respectés.
- ☐ Une console de jeu doit tourner sous système à temps réel « hard », sans quoi il est impossible de faire tourner un jeu.
- ☐ La tête d'injection d'une imprimante doit tourner sous système à temps réel « hard », sans quoi la page imprimée peut être erronée.
- ☐ La sonde de mesure de vitesse d'un avion doit tourner sous système à temps réel « hard », sans quoi le pilote automatique peut se désactiver.
- ☐ Un système à temps réel « soft » est plus rapide qu'un système à temps réel « hard ».
- ☐ Un système à temps réel « hard » est plus rapide qu'un système à temps réel « soft ».
- ☐ Windows 10 est un OS capable de travailler en temps réel « hard ».
- ☐ Ubuntu Desktop est un OS capable de travailler en temps réel « hard ».
- ☐ RTLinux est un OS capable de travailler en temps réel « hard ».

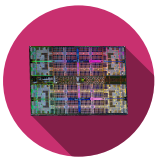
isa/riscv-to-machinecode-04

1.12 Type U vers code machine

Encoder l'assembleur RISC-V suivant en code machine.

```
lui s5, 0x8CDEF
```

isa/riscv-to-machinecode-05



1.13 Type J vers code machine

Encodez la **première** instruction de saut de l'assembleur RISC-V en code machine. Le programme est le suivant :

```
0x0000540C        jal ra,func1 # <--
0x00005410        add s1, s2, s3
...
0x001ABC04 func1: add s4, s5, s8
...
```

isa/riscv-to-machinecode-06

2 | Complément au laboratoire

Pour vous aider, n'hésitez pas à utiliser l'interpréteur RISC-V sur https://course.hevs.io/car/riscv-interpreter/ ainsi que Ripes.

Attention aux types des variables !

- Le type **int** est considéré de taille 32 bits signé.
- Le type **unsigned int** est considéré de taille 32 bits non-signé.
- Si il est suivi d'un nombre (ex: **int16_t**), cela signifie que la variable est sur x bits (ici 16). Si précédé d'un **u**, il est non-signé.

uint8_t est donc un byte non-signé, tandis que **int8_t** est un byte signé.

2.1 Calculs de base

a)

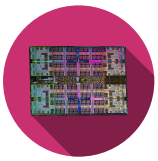
```
int b = 1;
int c = 2;
a = b + c;

int b = -1;
int c = 2;
a = b + c;

int b = -12;
int c = 2023;
a = b + c;
```

b)

```
int b = 2;
int c = 3;
int e = -1;
int f = -78;
int g = 2023;
int h = -12;
a = b - c;
d = (e + f) - (g + h);
```

isa/lab-basic-calc

2.2 Accès mémoire



```c
uint16_t a = mem[3];
mem[4] = a;

int16_t a = mem[3];
mem[4] = a;
```

isa/lab-memory

2.3 Algorithmes basiques

1. Transmettre la valeur de 8 bits de la mémoire à l'adresse 0x0000′1000 en série, bit par bit, dans le LSB de la mémoire à l'adresse 0x0000′1001. Les bits restants de l'adresse mémoire 0x0000′1001 doivent être ‹ **0** ›. Calculez le débit de bauds en $\frac{\text{Instructions}}{\text{Bit}}$ pour l'ensemble de la transmission.

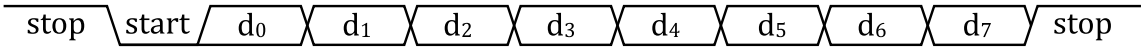


Fig. 7. – Transmission sérielle UART

2. Multiplier deux nombres de 4 bits ensemble en utilisant en plus une des commandes **bne** ou **bge**. L'algorithme fonctionne de la manière suivante : une multiplication est la même chose que l'addition x fois du même nombre. Par exemple : $2 * 9 = 9 + 9 = 18$.

isa/lab-basic-algos

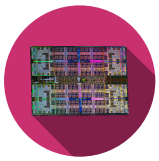
2.4 Branching

2.4.1 If / else

```c
int a = 1, b = 2, c;

if(a == b) {
  c = 0;
} else if(a > b) {
  c = 1;
} else {
  c = 2;
}
```

2.4.2 Switch case

```c
int a;

switch(mem[2]) {
  case 0:
    a = 17;
    break;
  case 3:
    a = 33;
    break;
  case 8:
  case 12:
    a = 10;
    break;
  default:
    a = 99;
}
```



2.4.3 While / Do While

```c
// A
int a = 10;

do{a = a - 1;}
while(a != 0);

// B
int a = 10;

do{a = a - 1;}
while(a >= 0);

// C
unsigned int a = 10;

do{a = a - 1;}
while(a >= 0);
```

2.4.4 For

```c
int a = 0, i;

for(i = 4; i > mem[0]; i = i - 1) {
  a = a + i;
}
```

isa/lab-branch

2.5 Functions

a)

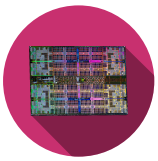
```c
int a = 1, b;
b = doubleIt(a);
b = doubleItOpti(a);

...

// Non-optimized version
// Let's assume a is saved in s0
int doubleIt(int myvar) {
  int a = myvar; // we WANT a in s0 !
  a = a * 2;
  return a;
}

// Optimized version
// Choose your registers freely
// Try having the less possible
//   instructions
int doubleItOpti(int myvar) {
  int a = myvar; //
  a = a * 2;
  return a;
}
```

b)

```c
int a=1, b=2, c=3,
 d=4, e=5, f=6, g=7,
 h=8, i=9, j=10, res;
res = sum(a,b,c,d,e,f,
  g,h,i,j);

...

int sum(int v1, int v2,
  int v3, int v4, int v5,
  int v6, int v7, int v8,
  int v9, int v10){
  int c;
  c = v1 + v2 + v3 + v4 +
    v5 + v6 + v7 + v8 +
    + v9 + v10;
  return c;
}
```

isa/lab-fcts

2.6 Advanced Algorithmus



2.6.1 Modulo

Le modulo % est une opération qui se pratique sur deux nombres entiers positifs et n'est autre que le reste de la division. Par exemple, 5 divisé par 3 donne 1 (on peut faire passer une fois 3 dans 5), **reste 2**.
Le modulo d'un nombre par 0 n'est pas défini.
La définition pour les nombres signés diverge selon le langage. Nous ne traitons que la version avec les unsigned.
Le modulo a plusieurs utilités, permettant de plafonner des valeurs, extraire de l'information, calculer une position X et Y à partir d'une valeur X*Y dans un tableau de taille connue ...

- Donner un code permettant d'effectuer cette opération pour n'importe quel entier positif en utilisant le set RV32IM.
- Comment la même chose peut-elle être implémentée dans RV32I ? Décrivez le(s) concept(s).

Le rôle du compilateur est d'optimiser au mieux le code. Si l'opération détectée est un modulo avec une constante étant une puissance de 2 (ex. x % 2, y % 8 ...), une variante n'incluant aucune division est possible.

- Donner cette variante.

> La notion de modulo pour les nombres réels est arrivée avec l'évolution des puissances de calcul et le résultat diverge aussi selon le langage. En C, une fonction spécifique des librairies std est nécessaire, **fmod()**. En Python, cette opération est native. Dans tous le cas, elles sont plus gourmandes en ressources et nécessitent de gérer certains cas spécifiques (NaN, infinity).

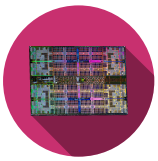
2.6.2 °F -> °C

Nous souhaitons créer une fonction capable de convertir les degrés Fahrenheit en Celsius. Comme les valeurs Fahrenheit varient entre 32 et 1000, une précision au degré près est suffisante.
La formule est simple : $C = (F - 32) * \frac{5}{9}$.
Ce serait pratique si un FPU était disponible, mais seul le jeu d'instructions de base RV32I est supporté.
Pour contourner ce problème, vous pouvez utiliser quelques astuces dont l'algorithme est le suivant :

A. Calculer $C = F - 32$.
B. Multiplier par 5
C. Diviser par 9
   - $\frac{1}{9}$ peut être enregistré par une représentation binaire spéciale

```
+-----+-----+-----+-----+-----+-----+------------+------------+
| b31 | b30 | b29 | b28 | b27 | ... |         b1 |         b0 |
+-----+-----+-----+-----+-----+-----+------------+------------+
| 2^0 | 2^-1| 2^-2| 2^-3| 2^-4| ... |      2^-30 |      2^-31 |
+-----+-----+-----+-----+-----+-----+------------+------------+
| 1   | 1/2 | 1/4 | 1/8 | 1/16| ... |1/1737418240|1/2147483648|
+-----+-----+-----+-----+-----+-----+------------+------------+
```

   dans ce cas, il s'agit de $0000\_1110\_0011\_1000\_1110\_0011\_1000\_1110_2$.



- La constante peut être précalculée et vaut $\frac{2^n}{9} + 1$. Plus n est grand, plus la précision est élevée. Le nombre de bits définit la taille maximale de n. Le $+1$ est un arrondi pour la précision perdue.
- Prenons $n = 16$. Notre nombre magique est donc magic $= \frac{2^n}{9} + 1 = \frac{65536}{9} + 1 = 7282$.
- Il faut multiplier la valeur par ce nombre magique

D. ensuite le diviser par 2^n. Dans le cas $n = 16 \rightarrow \frac{1}{65536}$.

Pour simplifier le travail, plusieurs hypothèses sont faites :
- Le nombre magique et la température en Fahrenheit sont toujours positifs.
- La taille de la plus grande multiplication est :

$$\begin{aligned} \text{nbBits}_{\text{max}_{\text{fahrenheit}}} + \text{nbBits}_{\text{mult5}} + \text{nbBits}_{\text{magicNumber}} &= \\ 10(\text{max. 1000-32}) + 3 + (n - \text{nbBits}_{\text{div9}} + 1) &= \\ 10 + 3 + (16 - 4 + 1) &= \\ = 26 \text{ bits} & \end{aligned} \tag{1}$$

  - Elle ne dépasse jamais 32 bits pour $n < 23$.

Testez et optimiser la fonction :
- Ecrire le code correspondant.
- Tester avec différentes valeurs de Fahrenheit.
- Tester avec plusieurs valeurs pour n (16, 18, 20). *N'oubliez pas de recalculer le nombre magique.*
- Tester la fonction avec $n = [22, 23]$, pour $°F = [100, 400, 1000]$.

isa/lab-adv-algos

1 Architecture

1.1 Stack-Architecture

Évaluez l'expression $\frac{a+bc}{a+dc-e}$ en utilisant une pile d'évaluation de processeur.

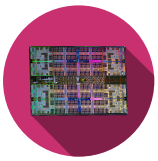
a) Écrire pseudo code du calcul.
b) Combien de références mémoire directes et indirectes sont nécessaires pour une taille de pile infini ?
c) Combien de références mémoire directes et indirectes sont nécessaires si la taille de la pile est de 2 ?

arc/stack-01

1.2 Stack-Architecture

Évaluez l'expression $\frac{(a+b)^2}{\pi} * (a + b + c)$ en utilisant une pile d'évaluation de processeur.

a) Écrire pseudo code du calcul.
b) Combien de références mémoire directes et indirectes sont nécessaires pour une taille de pile de 4 ?
c) Combien de références mémoire directes et indirectes sont nécessaires dans le cas d'une taille de pile de 3 ?
d) Combien de références mémoire directes et indirectes sont nécessaires si la taille de la pile est de 2 ?

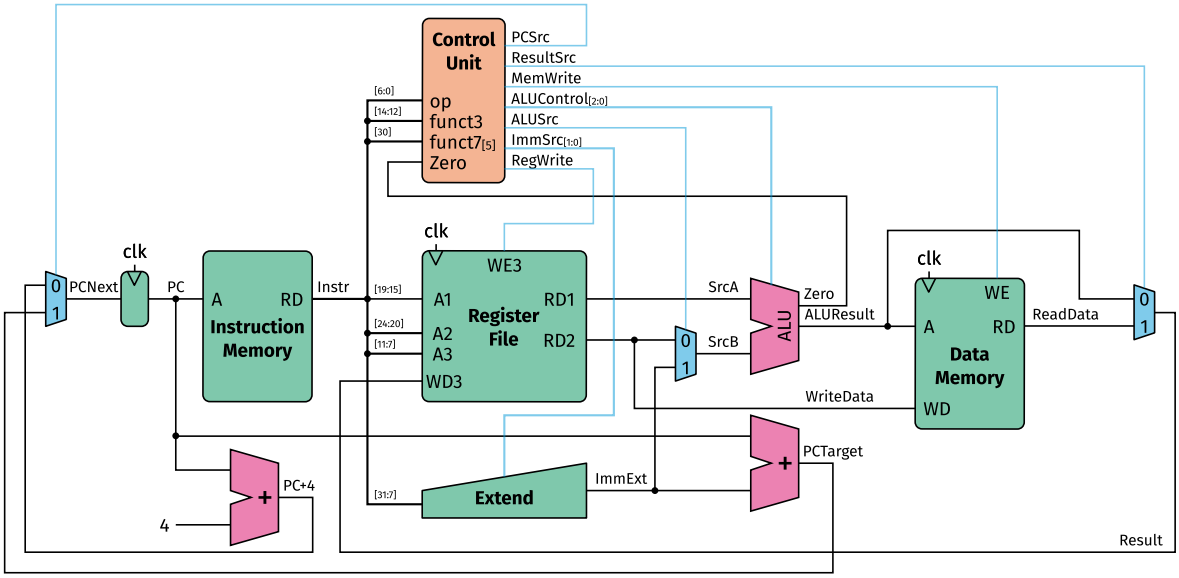


arc/stack-02

2 | Single-Cycle RISC-V

2.1 Fonctionnement du processeur à cycle unique

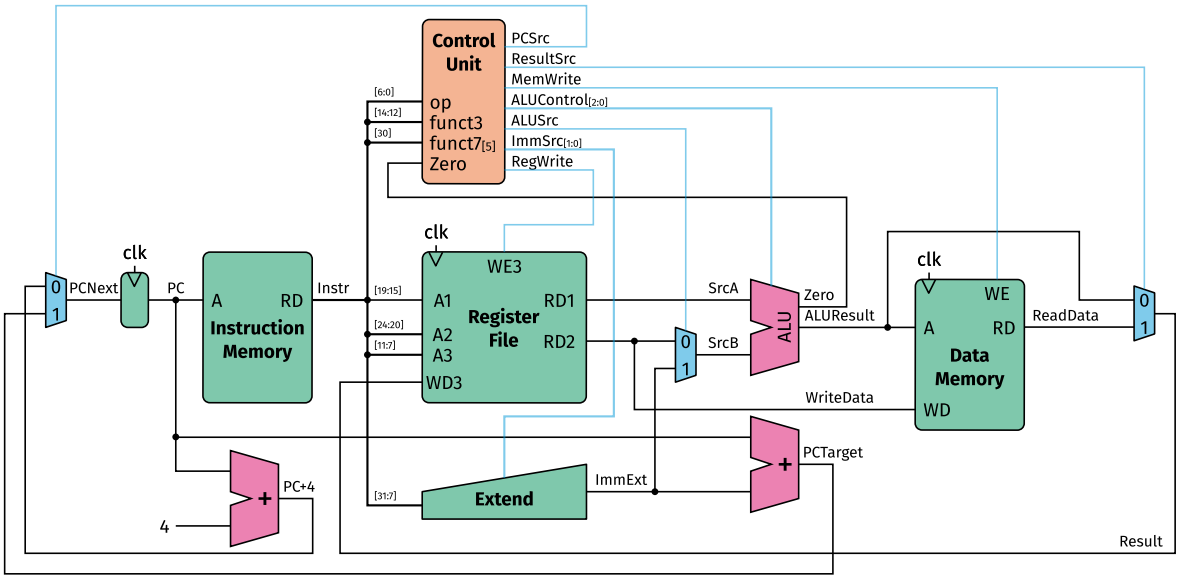
Déterminez les valeurs des signaux de commande et les portions du chemin de données qui sont utilisées lors de l'exécution d'une instruction **and**. Dessinez directement sur l'image le fonctionnement interne du processeur.



arc/scr-01

2.2 Prolonger le mono-cycle avec l'instruction `jal`

Montrez comment modifier le processeur RISC-V à mono-cycle donné pour prendre en charge l'instruction de saut et de liaison **jal**. **jal** écrit **PC+4** dans **rd** et modifie le PC à l'adresse cible du saut **PC+imm**.



arc/scr-02

2.3 Performance du processeur à mono-cycle

Un processeur mono-cycle construit avec un processus de fabrication CMOS de 7 nm présente les caractéristiques de timing suivantes.

Element	Parameter	Delay(ps)
Register clk-to-Q	t_{pcq}	40
Register Setup	t_{setup}	50
Multiplexer	t_{mux}	30
AND-OR Gate	$t_{\text{AND\_OR}}$	20
ALU	t_{ALU}	120
Decoder (Control Unit)	t_{dec}	25
Extend Unit	t_{ext}	35
Memory Read	t_{mem}	200
Register File Read	t_{RFread}	100
Register File Setup	t_{RFSetup}	60

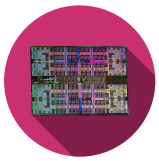
Le programme du benchmark SPECINT2000 contient 100 milliards d'instructions.
Calculez le temps d'exécution du benchmark pour ce processeur à mono-cycle.

arc/scr-03

3 Multi-Cycle RISC-V

3.1 Performance du processeur à multi-cycle

Le programme du benchmark SPECINT2000 se compose d'environ 25% loads, 10% stores, 11% branches, 2% jumps, et 52% R- ou I-Type ALU Instructions.



Déterminez le CPI moyen pour ce benchmark pour le processeur multi-cycle que nous avons développé.

arc/mcr-01

3.2 Performance du processeur à multi-cycle

Un processeur multi-cycle construit avec un processus de fabrication CMOS de 7 nm présente les caractéristiques de timing suivantes.

Element	Parameter	Delay(ps)
Register clk-to-Q	t_{pc}	40
Register Setup	t_{setup}	50
Multiplexer	t_{mux}	30
AND-OR Gate	$t_{\text{AND\_OR}}$	20
ALU	t_{ALU}	120
Decoder (Control Unit)	t_{dec}	25
Extend Unit	t_{ext}	35
Memory Read	t_{mem}	200
Register File Read	t_{RFread}	100
Register File Setup	t_{RFSetup}	60

Le programme du benchmark SPECINT2000 contient 100 milliards d'instructions. Utilisez le CPI_{avg} de la tâche précédente.
Calculez le temps d'exécution du benchmark pour ce processeur à multi-cycle.

arc/mcr-02