

Test Microprocessors Architecture Configuration

Javier Andres Tarazona Jimenez
Ingénieur Degree Programme
STIC
ENSTA Paris
Paris, France
javier-andres.tarazona@
ensta-paris.fr

Jair Anderson Vasquez Torres
Ingénieur Degree Programme
STIC
ENSTA Paris
Paris, France
jair-anderson.vasquez@
ensta-paris.fr

Maeva
Ingénieur Degree Programme
STIC
ENSTA Paris
Paris, France
maeva@ensta-paris.fr

Carlos
Ingénieur Degree Programme
STIC
ENSTA Paris
Paris, France
carlos@ensta-paris.fr

Abstract—...
Index Terms—...

I. EXERCICE 4 — PROFILING (Q1)

a) *Objectif.*: Le *profiling* de l’instruction *mix* consiste à mesurer la répartition des instructions exécutées par grandes catégories (calcul entier, mémoire, contrôle, etc.). Cette information est essentielle en architecture : elle permet d’identifier où se situe la *pression dominante* (unités de calcul, hiérarchie mémoire, prédiction de branchements), et donc d’anticiper quels leviers microarchitecturaux sont les plus pertinents.

b) *Méthode.*: Nous avons simulé les exécutions avec gem5 en mode SE (ISA RISC-V), en configuration de type A7 (se_A7.py). Les compteurs proviennent des instructions *committed* (retired). Le nombre total d’instructions exécutées est $N = \text{simInsts}$. Pour chaque catégorie c , on note I_c le nombre d’instructions appartenant à c . Le pourcentage associé est :

$$\text{pct}(c) = 100 \times \frac{I_c}{N}.$$

Les valeurs sont arrondies à deux décimales (somme $\approx 100\%$).

TABLE I
RÉPARTITION DES INSTRUCTIONS EXÉCUTÉES (PROFILING) —
CONFIGURATION A7.

Catégorie	Dijkstra (A7)		Blowfish (A7)	
	Count	%	Count	%
ALU entier	22118141	44.09%	1882443	55.42%
Chargements (Load)	11799269	23.52%	771841	22.72%
Stockages (Store)	4853941	9.68%	408487	12.03%
Branches (contrôle)	9870255	19.67%	334128	9.84%
Mult/Div entier	1517371	3.02%	6	0.00%
Flottant (FP)	12	0.00%	12	0.00%
Autres	10476	0.02%	18	0.00%
TOTAL (simInsts)	50169465	100.00%	3396935	100.00%

c) *Résultats détaillés.*:

d) *Synthèse par familles (lecture plus “architecture”).*: Afin de mieux comparer les pressions microarchitecturales, on regroupe les catégories en trois familles : *calcul entier* (ALU + Mult/Div), *mémoire* (Load + Store) et *contrôle* (Branches).

TABLE II
AGRÉGATION PAR FAMILLES : CALCUL, MÉMOIRE ET CONTRÔLE (A7).

Famille	Dijkstra (%)	Blowfish (%)	Δ (BF – Dij) [points]
Calcul entier (ALU + Mult/Div)	47.11	55.42	+8.31
Mémoire (Load + Store)	33.20	34.75	+1.55
Contrôle (Branches)	19.67	9.84	-9.83
Reste (FP + Autres)	≈ 0.02	≈ 0.00	≈ 0

e) *Interprétation (comparaison chiffrée).*: Les deux applications présentent une part mémoire comparable ($\approx 33.20\%$ pour Dijkstra contre $\approx 34.75\%$ pour Blowfish), ce qui indique que la hiérarchie mémoire (L1/L2) reste un levier important dans les deux cas. En revanche, Dijkstra est nettement plus *control-heavy* : les branches représentent 19.67% des instructions contre 9.84% pour Blowfish, soit environ $\times 2$ de densité de contrôle. À l’inverse, Blowfish est davantage *compute-heavy* : la part d’ALU entier atteint 55.42% contre 44.09% pour Dijkstra (+11.33 points). Les instructions FP sont négligeables (quelques occurrences attribuables à un surcoût du runtime plutôt qu’au noyau algorithmique).

A. Exercice 4 — Q2 : Catégorie à améliorer

Sur **Dijkstra**, la part **contrôle** est élevée (**branches = 19.67%**) et s’ajoute à une pression **mémoire** déjà importante (**load+store = 23.52% + 9.68% = 33.20%**) : améliorer la **prédiction de branchements** (et réduire les bulles/flush) est donc un levier prioritaire pour ce code très *branchy*. Sur **Blowfish**, le profil est surtout **compute-heavy** en entier (ALU = **55.42%** avec **branches = 9.84%**) : l’amélioration la plus pertinente concerne le **débit/latence des opérations entières** (ALU), tandis que la mémoire reste secondaire bien que non

négligeable (**load+store = 34.75%**). Ainsi, la catégorie à optimiser dépend de l'application : **contrôle (branches) pour Dijkstra** et **calcul entier (ALU) pour Blowfish**.

REFERENCES

- [1] A. Huamán, "Feature Description," *OpenCV Documentation* (OpenCV 4.14.0-pre), accessed Feb. 6, 2026. [Online]. Available: https://docs.opencv.org/4.x/d5/dde/tutorial_feature_description.html