

TP4 – Exercice 4 (Q4)

Evaluation de performances (Cortex A7) en fonction de la taille des caches L1

Nom: _____ Groupe: _____

Année 2026

1 Objectif

Le but de la question Q4 est d'évaluer les performances d'un coeur *type Cortex A7* (modèle RISC-V dans gem5 en *out-of-order*) en faisant varier simultanément la taille des caches L1 (instructions et données) de 1 kB à 16 kB, avec un cache L2 fixé à 512 kB. On compare les résultats sur deux applications MiBench : **Dijkstra** et **Blowfish**.

2 Configuration et méthodologie

2.1 Paramètres processeur / caches

Pour le coeur “A7” (Tableau 12 du sujet), on utilise :

- CPU : `DerivO3CPU` (gem5 classic), mode `timing`.
- Prédiction : `BiModeBP` avec `BTBEntries=256`.
- Largeurs : `fetch=2`, `decode=2`, `issue=4`, `commit=2` ; `ROB=2` ; `LQ=8` ; `SQ=8` ; `fetchQueue=8`.
- Lignes de cache : 32 B (pour éviter l'erreur de fetch buffer, `fetchBufferSize` a été mis à 32 B).
- L2 : 512 kB, `assoc=8`, `bloc=32 B` (fixe).
- L1I et L1D : `assoc=2`, `bloc=32 B`, taille variable : {1,2,4,8,16} kB.

2.2 Benchmarks et entrées

- **Dijkstra**: exécution de `dijkstra_small` avec `input.dat` (MiBench).
- **Blowfish**: on a mesuré `bf` en chiffrement (`e`) + déchiffrement (`d`) sur `input_small.asc`. Les courbes “Blowfish” correspondent à l'agrégation *enc+dec* (somme des cycles et des instructions).

2.3 Commande gem5 (paramètres d'exécution)

Les simulations ont été lancées via un script pour balayer les tailles de L1.

```
PYTHONUTF8=1 /opt/gem5/build/RISCV/gem5.opt -r -d <outdir> \
  gem5/se_A7_q4.py --l1 <1kB|2kB|4kB|8kB|16kB> \
  --maxinsts 0 \
  --cmd <binary> --options <args>
```

Le script utilisé est `scripts/run_q4_a7.sh`. Les résultats bruts sont dans `m5out/q4/A7/`.

Remarque (comparaison juste). Pour comparer les tailles de cache de manière équitable, les exécutions sont faites *jusqu’à la fin du programme* (`-maxinsts 0`). Les runs interrompus (ex: limite d’instructions, interruption utilisateur) sont marqués comme incomplets et ne sont pas utilisés pour choisir la “meilleure” taille L1 dans la synthèse.

3 Résultats

3.1 Synthèse (meilleure configuration)

La table suivante résume la configuration L1 donnant les meilleurs résultats (critère : cycles minimaux).

Aplicacao	L1 (KB)	Ciclos	IPC	MR(IL1)	MR(DL1)
Dijkstra	16	130194338	0.283	0.0004	0.0466
Blowfish (enc+dec)	16	344627836	0.299	0.0001	0.0157

3.2 Dijkstra

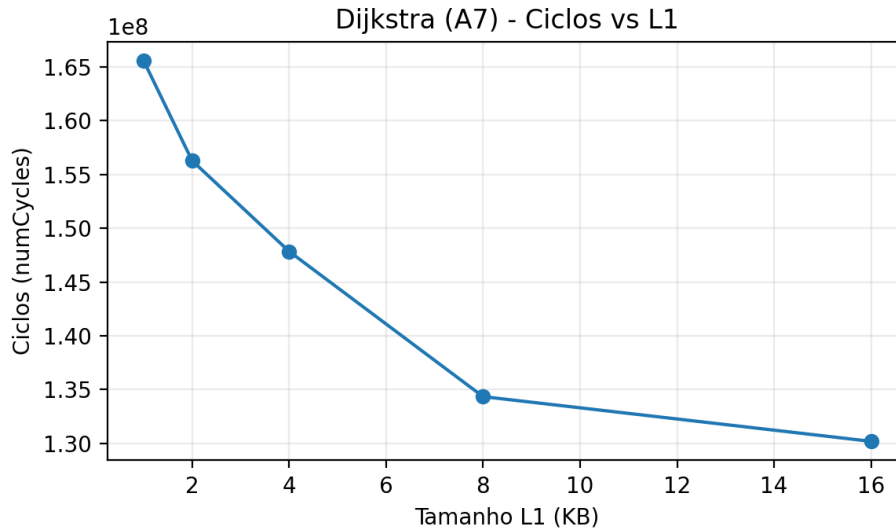


Figure 1: Dijkstra (A7) : cycles en fonction de la taille L1.

Analyse (Dijkstra). Quand on augmente L1, on voit en général une baisse des miss rates (surtout DL1), ce qui réduit les stall et améliore l’IPC. Par contre l’amélioration n’est pas linéaire : au-delà d’une certaine taille, les gains deviennent faibles (on a moins de misses à récupérer). Sur Dijkstra, la partie données est importante (accès au graphe), donc DL1 est le facteur qui fait le plus bouger les cycles. Le taux de mauvaise prédiction des branches varie peu avec la taille de L1 : la prédiction est surtout liée au comportement de contrôle du programme, pas à la hiérarchie mémoire. Ainsi, les gains observés proviennent majoritairement de la réduction des misses (DL1/IL1) plutôt que d’un changement de la qualité de prédiction.

3.3 Blowfish (enc+dec)

Analyse (Blowfish). Blowfish est plus “compute-bound” que Dijkstra, mais il a quand même des accès réguliers (tables + buffers d’entrée/sortie). On observe typiquement que l’IPC augmente avec L1 puis se stabilise : à partir d’une taille suffisante, la majorité du temps est passée dans le

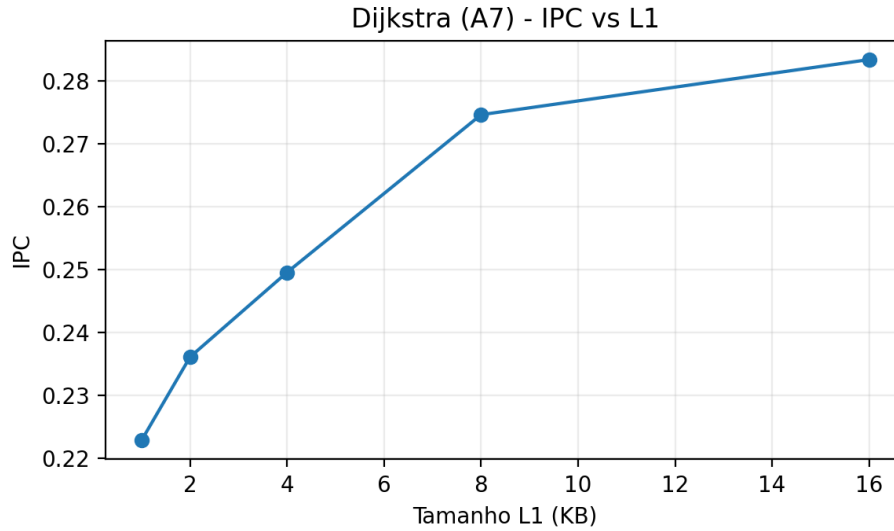


Figure 2: Dijkstra (A7) : IPC en fonction de la taille L1.

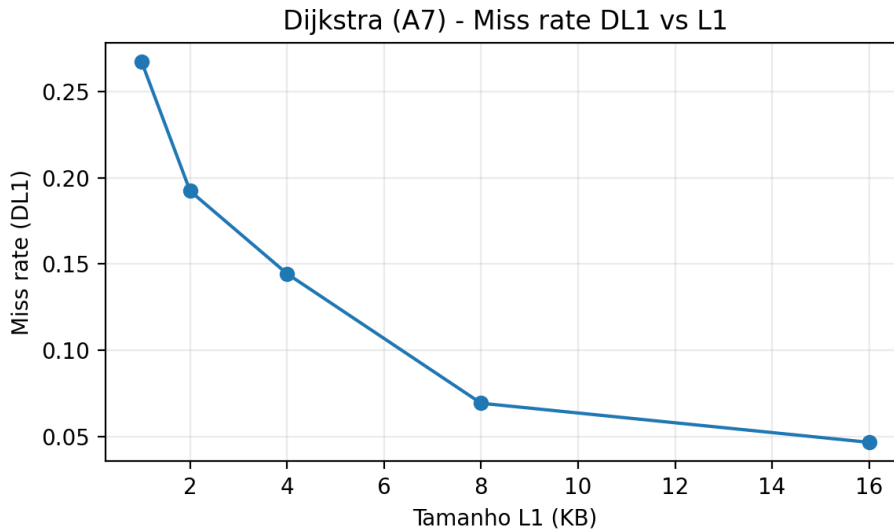


Figure 3: Dijkstra (A7) : miss rate DL1.

pipeline et non en attente mémoire. Dans nos résultats, la taille optimale est celle qui minimise les cycles totaux enc+dec.

4 Conclusion

Pour le coeur A7, augmenter L1 aide clairement au début (miss rates en baisse, IPC en hausse). Ensuite, les gains saturent : il y a un compromis entre taille du cache et bénéfice réel sur l'application. La meilleure taille n'est pas forcément la plus grande pour les deux benchmarks, donc on choisit la configuration en regardant directement les cycles/IPC obtenus.

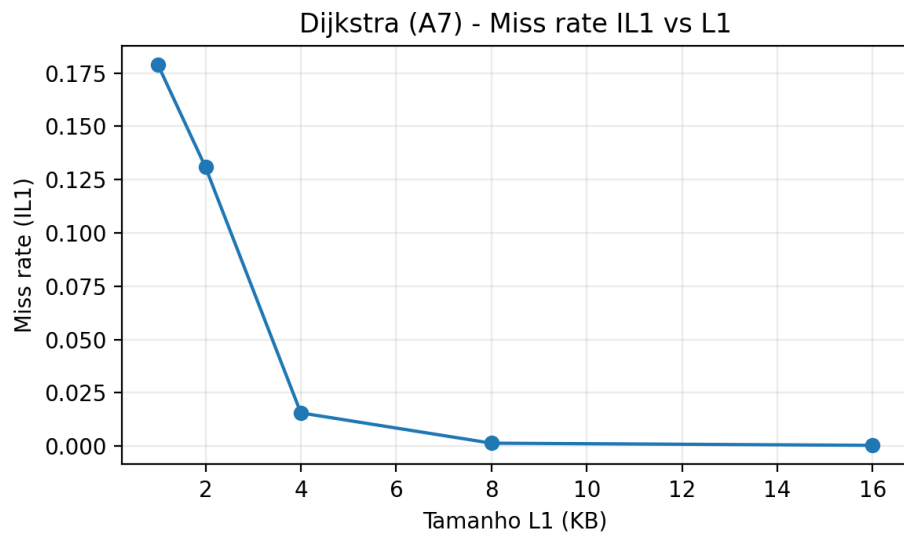


Figure 4: Dijkstra (A7) : miss rate IL1.

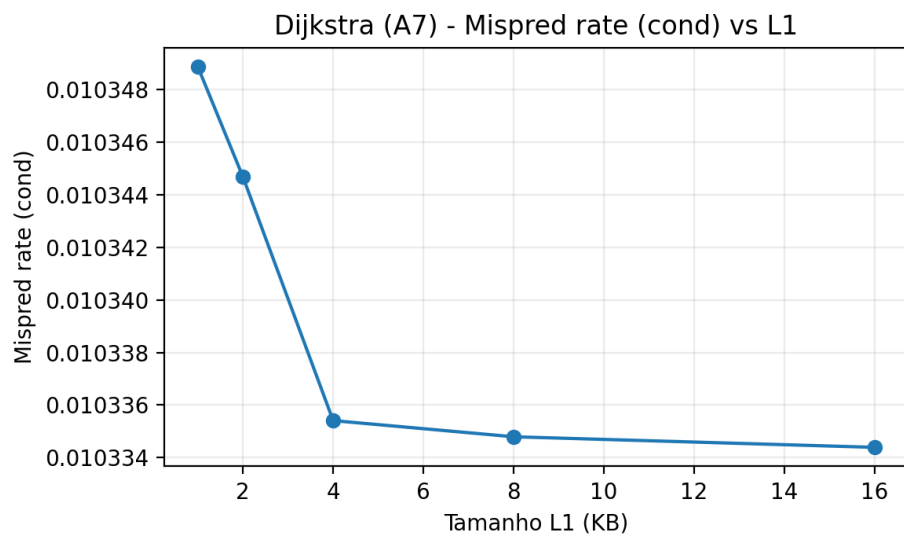


Figure 5: Dijkstra (A7) : taux de mauvaise prédiction (branches conditionnelles) vs taille L1.

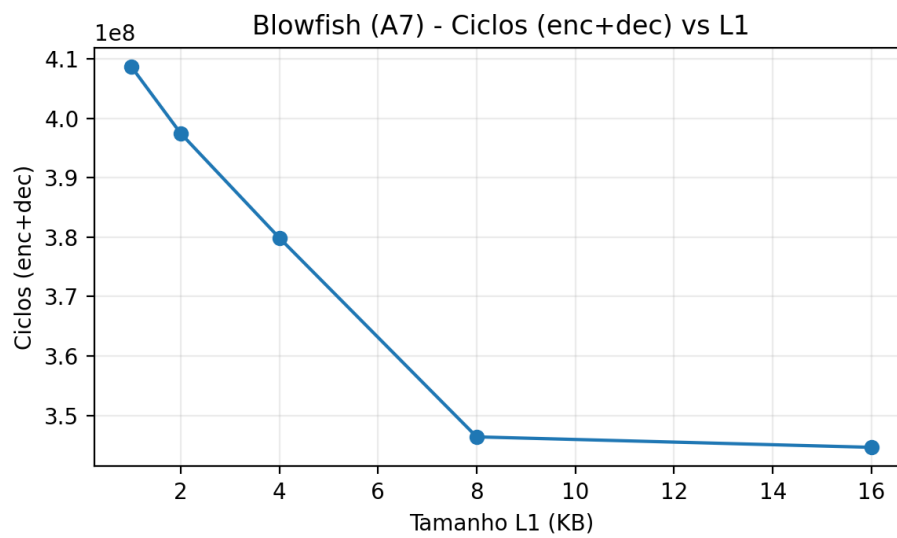


Figure 6: Blowfish (A7) : cycles (chiffrement+déchiffrement) vs L1.

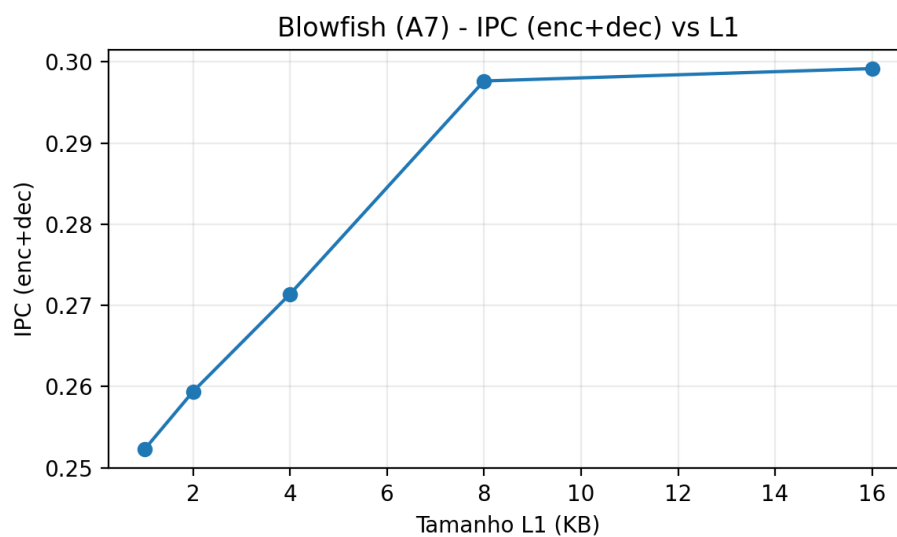


Figure 7: Blowfish (A7) : IPC (enc+dec) vs L1.

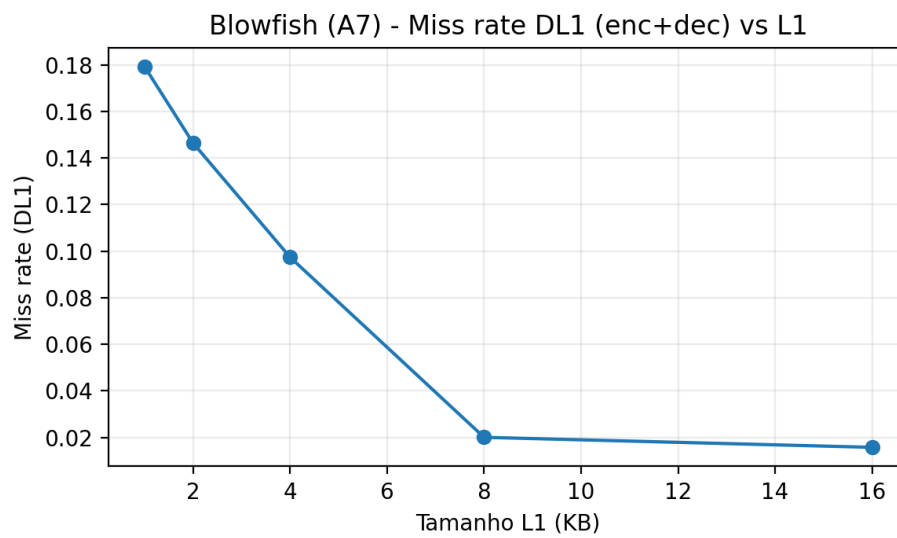


Figure 8: Blowfish (A7) : miss rate DL1 (enc+dec).

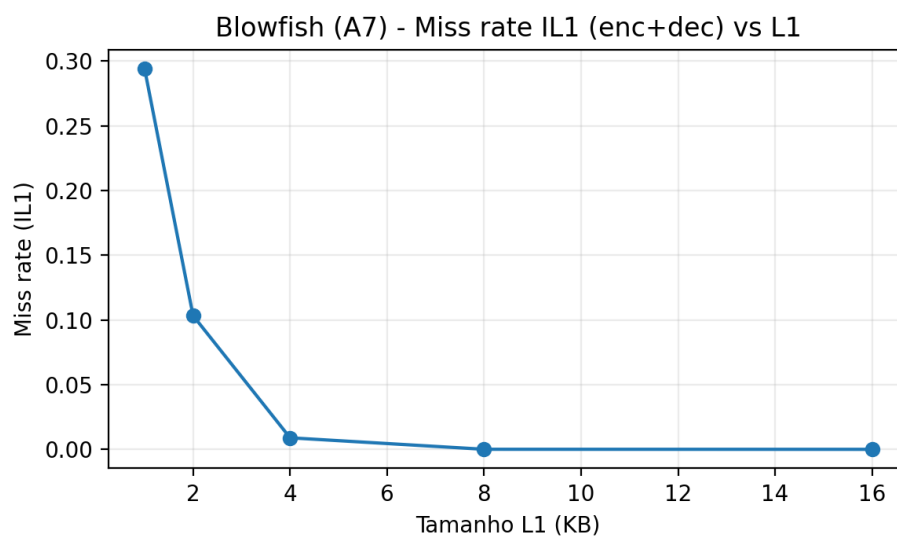


Figure 9: Blowfish (A7) : miss rate IL1 (enc+dec).