ES201

Mémoires caches - Evaluation des performances de différentes configurations de mémoires caches

Agathe BEUCHER, Magali CASAMAYOU, Cyprien GAUTHIER et Ethan ABIMELECH

Table des matières

Table des figures

1 Description du problème

Dans ce TP, nous allons analyser les performances de deux cœurs ARM embarqués : un Cortex A15 (hautes performances) et un Cortex A7 (haute efficacité énergétique), sur des applications type traitement de graphes.

2 Profiling de l'application

⟶ Objectif : identifier la classe des instructions sollicitées et le pourcentage d'instructions exécutées issues de cette classe afin d'optimiser le nombre d'unités fonctionnelles correspondant à ce type.

2.1 Question 1

Classes d'instructions	**Blowfish**(%)	**Dijkstra**(%)
Instructions de lecture (Load)	8.12	26.94
Instructions d'écriture (Store)	**45.61**	7.92
Instruction de branchement inconditionnel (Uncond branch)	2.18	0.80
Instruction de branchement conditionnel (Cond branch)	11.37	17.09
Instruction de calcul entier (Int computation)	32.62	**47.25**
Instruction de calcul flottant (Fp computation)	0.00	0.00

TABLE 1 – **Tableau des pourcentages de chaque classe d'instructions par Benchmark**

En utilisant l'option "*-iclass*" lors du profiling de chaque application, on obtient immédiatement le pourcentage d'instructions issues de chaque classe.

2.2 Question 2

On doit à priori s'intéresser aux catégories ayant le plus fort pourcentage d'instructions exécutées pour saisir lesquelles nécessitent une amélioration de performance.

- Pour *Dijkstra*, il faudra regarder la classe **Instructions d'écriture** (47.25%)
- Pour *Blowfish* la classe **Instruction de calcul entier** (53.73%) mérite une amélioration de performance

2.3 Question 3

D'après le TP2,

Classes d'instructions	**Blowfish**(%)	**Dijkstra**(%)	**Produit**(%)	**SSCA2-BCS**(%)	**SHA-1**(%)
Load	8.12	26.94	13.30	**37.11**	16.72
Store	**45.61**	7.92	6.28	9.23	8.12
Uncond branch	2.18	0.80	7.96	5.95	8.08
Cond branch	11.37	17.09	16.13	6.65	6.10
Int computation	32.62	**47.25**	**45.76**	**37.62**	**68.99**
Fp computation	0.00	0.00	10.56	3.44	0.00

TABLE 2 – **Tableau des pourcentages de chaque classe d'instructions par Benchmark**

On peut remarquuer que Dijkstra se distingue des autres benchmark par des pourcentages relativement élevés dans les classes d'instructions de branchement et de calcul entier, tandis que Blowfish montre une tendance plus marquée vers les instructions de stockage. SSCA2-BCS se concentre également sur les calculs entiers. A l'instar de Dijkstra, le produit de polynômes semble avoir un besoin significatif de de branchement et de calcul entier, mais auxquelles s'ajoute un besoin de calculs en flottants. SHA-1 se distingue par un besoin important de calculs entiers, justifié par sa nature d'algorithmes de hachage cryptographique. Il effectue également des opérations de chargement et de stockage de données, mais dans une moindre mesure que certaines autres classes d'instructions.

3 Evaluation des performances

3.1 Question 4

3.1.1 Fonctionnement des paramètres de sim-outorder

Par exemple pour configurer l'instruction cache **il1 :128 :32 :1 :l**, on a :

- 128 : nombre d'ensemble dans le cache : Taille du cache / (Taille d'un bloc * Associativité)
- 32 : taille d'une ligne en octet (bloc)
- 1 : degré d'associativité du cache
- l : indicateur de latence du cache

On considère le Cortex-A7, donc pour

1. **Instruction cache et Data cache Cortex-A7**

Paramètre	**Valeur**
cache size (KB)	[1,2,4,8,16]
nb d'element dans le cache	[16,32,64,128,256]
bloc size (bytes)	32
associativity	2
IL1	**il1 :[nb_elem] :32 :2 :l**
DL1	**dl1 :[nb_elem] :32 :2 :l**

TABLE 3 – Paramètres de cache L1 Cortex-A7

2. **L2 cache Cortex-A7**

Paramètre	Valeur
cache size (KB)	512
nb d'element dans le cache	64
bloc size (bytes)	32
associativity	8
UL2	**ul2 :64 :32 :8 :l**

TABLE 4 – Paramètres de cache L2 Cortex-A7

De même pour le Cortex-A15, on a :

1. **Instruction cache et Data cache Cortex-A15**

Paramètre	Valeur
cache size (KB)	[2,4,8,16,32]
nb d'element dans le cache	[16,32,64,128,256]
bloc size (bytes)	64
associativity	2
IL1	**il1 :[nb_elem] :64 :2 :l**
DL1	**dl1 :[nb_elem] :64 :2 :l**

TABLE 5 – Paramètres de cache L1 Cortex-A15

2. **L2 cache Cortex-A15**

Paramètre	Valeur
cache size (KB)	512
nb d'element dans le cache	512
bloc size (bytes)	64
associativity	16
UL2	**ul2 :512 :64 :16 :l**

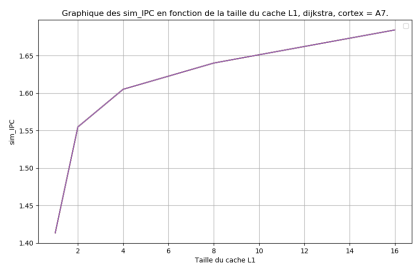
TABLE 6 – Paramètres de cache L2 Cortex-A15

On automatise les graphes (voir git : ***q4.py***). On trace pour chaque test dijkstra et blowfish :

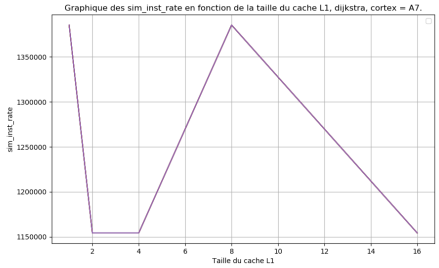
- "sim_inst_rate"
- "sim_IPC"
- "bpred_bimod.bpred_addr_rate"
- "bpred_bimod.bpred_dir_rate"
- "sim_elapsed_time"

On choisit ici par souci de mise en page de n'afficher que les graphes qui ont une réelle évolution en fonction de la taille des caches L1, mais l'ensemble des graphes est disponible sur le dépôt git.

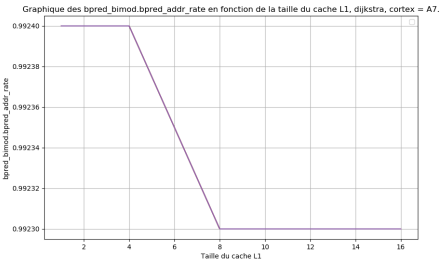
3.1.2 Dijkstra Cortex-A7



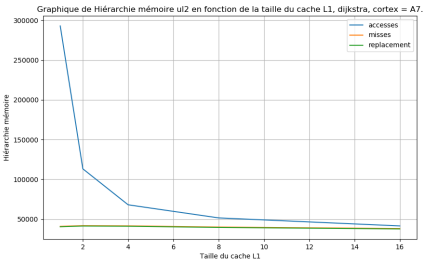
(a) Sim_IPC



(b) Sim_inst_rate



(c) bpred_bimod.bpred_addr_rate



(d) Hiérarchie mémoire ul2

FIGURE 1 – Performance de **Dijkstra** avec Cortex-A7

- **Taux d'instructions par cycle** (sim_IPC) : L'IPC augmente avec la taille du cache L1, ce qui est attendu car un cache plus grand peut réduire les ratés de cache et améliorer les performances. Le meilleur IPC est observé avec le plus grand cache L1 testé, donc pour taille_L1=16KB, il est en revanche plus faible pour 1,2,4,8 KB.

- **Taux d'instructions par seconde** (sim_inst_rate) : Le taux de simulation d'instructions fait référence aux nombre d'instructions qu'il peut traiter par unité de temps. Un taux élevé indique une performance élevée du processeur, ce qui signifie qu'il peut accomplir un plus grand nombre d'opérations dans un laps de temps donné, ce qui est généralement souhaitable pour améliorer les performances globales du système. D'après le graphe, les performance générales les plus élevés sont pour 1KB et 8KB. Ces données sont d'ailleurs complémentaires avec celle de 'sim_elapsed_time', représente la durée totale nécessaire pour exécuter un programme, et qui est minimale pour 1KB et 8KB.

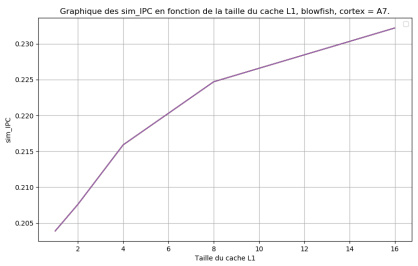
- **Taux de prédiction de branchement** (bpred_bimod.bpred_addr_rate et dir_rate) : Des taux de prédiction de branchement élevés sont préférables car ils réduisent le nombre de cycles de processeur gaspillés en raison de prédictions incorrectes. Les graphiques indiquent que ces taux sont les plus élevées pour des tailles de caches faibles, soit 1,2,4,8KB.

- **Hiérarchie mémoire dl1, il1, ul2** : On peut voir le nombre d'accès, de ratés et de remplacements pour les caches L1 et L2. Idéalement, on souhaiterait voir une diminution des misses et des remplacements à mesure que la taille du cache augmente, ce qui

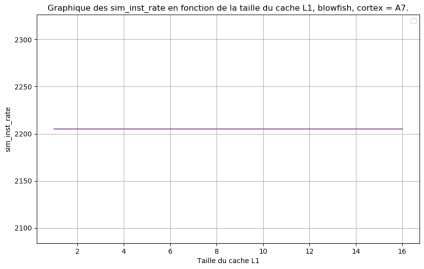
indiquerait une bonne utilisation de la mémoire cache. Cependant, dans notre les valeurs semblent rester relativement constantes pour il1, ce qui peut indiquer que la taille du cache L1 n'affecte pas significativement ces mesures pour le programme BlowFish ou que les caches sont suffisamment grands pour le jeu d'instructions utilisé. En revanche, pour les caches L2, pour une taille de cache l élevé (16KB) on a une bonne utilisation de la mémoire cache.En effet, un nombre moins élevé d'accès au cache L2 avec l'augmentation de la taille des caches L1 peut aussi signifier que le cache L1 fait son travail en filtrant les accès fréquents et en laissant seulement les accès moins fréquents au cache L2. Si les taux de raté du cache L2 deviennent bas, cela peut indiquer une hiérarchie mémoire efficace.

On a vu précédemment que Dijkstra se caractérise par pun fort besoin dans les classes d'instructions de branchement et de calcul entier, ainsi on ne peut pas négliger les taux de prédiction de branchement. Par ailleurs, il erait difficilement concevable de négliger également le taux d'instruction par seconde qui se doit d'être optimale. On peut ainsi considérer que pour une taille de cache **L1=1KB**, on a ces deux optimaux, malheureusement au détriment du nombre de taux d'instructions par cycle qui n'est pas optimisé...

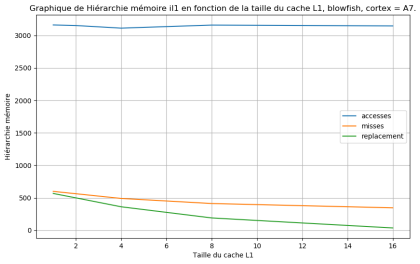
3.1.3 Blowfish Cortex-A7



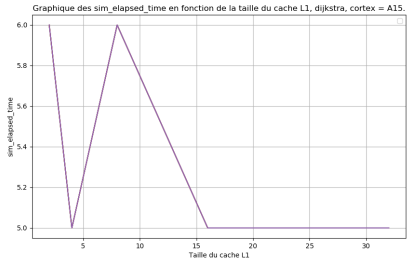
(a) Sim_IPC



(b) Sim_inst_rate



(c) Hiérarchie mémoire il1



(d) sim_elapsed_time

FIGURE 2 – Performance de **Blowfish** avec Cortex-A7

Analyse :

- **Taux d'instructions par cycle** (sim_IPC) : L'IPC augmente avec la taille du cache L1, ce qui est attendu car un cache plus grand peut réduire les ratés de cache et améliorer les performances. Le meilleur IPC est observé avec le plus grand cache L1 testé, donc pour taille_L1=16KB

- **Taux d'instructions par seconde** (sim_inst_rate) : Ce taux semble constant, indiquant que la vitesse d'exécution des instructions ne varient pas en fonction de la taille des caches L1

- **Taux de prédiction de branchement** (bpred_bimod.bpred_addr_rate dir_rate) : Des taux de prédiction de branchement élevés sont préférables car ils réduisent le nombre de cycles de processeur gaspillés en raison de prédictions incorrectes. Les graphiques indiquent que ces taux restent stables à travers les différentes tailles de cache, ce qui suggère que la prédiction de branchement n'est pas fortement influencée par la taille du cache L1 dans ce cas.

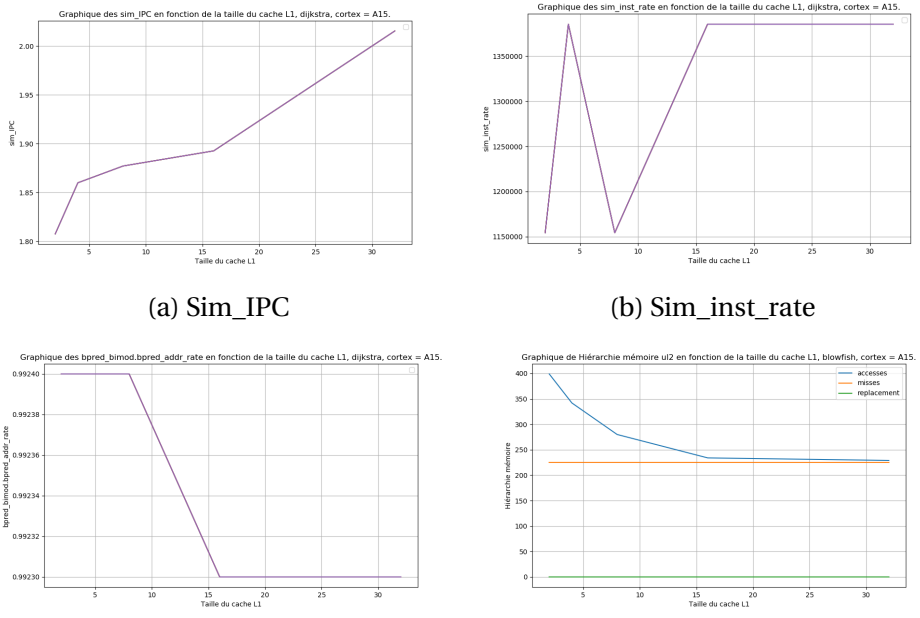
- **Hiérarchie mémoire dl1, il1, ul2** :De même que pour Dijkstra, les valeurs semblent rester relativement constantes pour il1, mais pour les caches L2, pour une taille de cache L1 élevé (16KB) donne une bonne utilisation de la mémoire cache.

- **Temps total** (sim_elapsed_time) : mesure du temps total écoulé pendant la simulation. Il représente la durée totale nécessaire pour exécuter un programme ou un ensemble d'instructions dans le cadre d'une simulation de performance de processeur

Ainsi, une taille de cache **L1=16KB** semble optimal pour Blowfish avec un Cortex-A7.

3.2 Question 5

3.2.1 Dijkstra Cortex-A15



(a) Sim_IPC

(b) Sim_inst_rate

(c) bpred_bimod.bpred_addr_rate

(d) Hiérarchie mémoire ul2

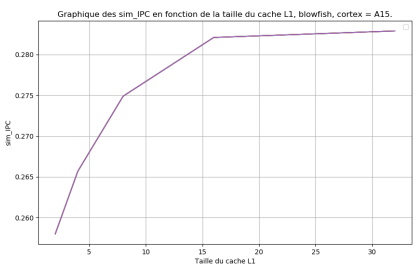
FIGURE 3 – Performance de **Dijkstra** avec Cortex-A15

Analyse :

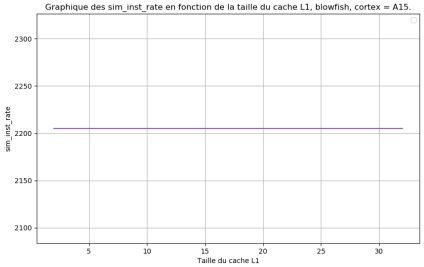
- **Taux d'instructions par cycle** (sim_IPC) : Le meilleur IPC est observé avec le plus grand cache L1 testé, donc pour taille_L1=16KB et croit linéairement à partir de 8KB.
- **Taux d'instructions par seconde** (sim_inst_rate) : les performance générales les plus élevés sont pour 4KB et à partir de 16KB
- **Taux de prédiction de branchement** (bpred_bimod.bpred_addr_rate et dir_rate) : Les graphiques indiquent que ces taux sont les plus élevées pour des tailles de caches faibles, soit 2,4,8KB.
- **Hiérarchie mémoire dl1, il1, ul2** : Pour les caches L2, pour une taille de cache L1 élevé (16KB) donne une bonne utilisation de la mémoire cache.

Ainsi, une taille de cache **L1=2KB** semble optimal pour Dijkstra avec un Cortex-A15.

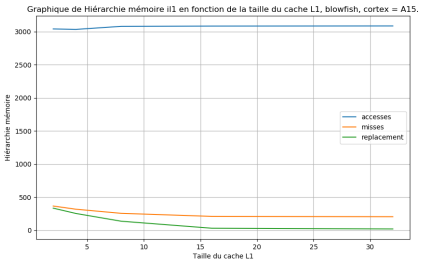
3.2.2 Blowfish Cortex-A15



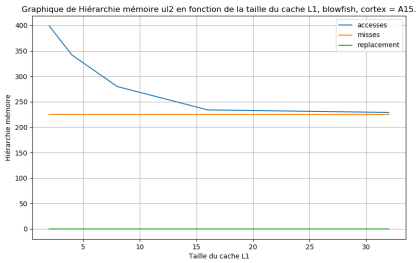
(a) Sim_IPC



(b) Sim_inst_rate



(c) Hiérarchie mémoire il1



(d) Hiérarchie mémoire ul2

FIGURE 4 – Performance de **Blowfish** avec Cortex-A15

- **Taux d'instructions par cycle** (sim_IPC) : Le meilleur IPC est observé avec le plus grand cache L1 testé, donc pour taille_L1=32KB
- **Taux d'instructions par seconde** (sim_inst_rate) : Ce taux semble constant, indiquant que la vitesse d'exécution des instructions ne varient pas en fonction de la taille des caches L1
- **Hiérarchie mémoire dl1, il1, ul2** :De même que pour Dijkstra, les valeurs semblent rester relativement constantes pour il1, mais pour les caches L2, pour une taille de cache L1 élevé (32KB) donne une bonne utilisation de la mémoire cache.

Ainsi, une taille de cache **L1=32KB** semble optimal pour Blowfish avec un Cortex-A15.

4 Efficacite surfacique

4.1 Question 6

En ouvrant le fichier cache.cfg, on peut déterminer les différents paramètres de cache :

```
# Cache size
//-size (bytes) 4096
//-size (bytes) 1048576
//-size (bytes) 2097152
//-size (bytes) 4194304
//-size (bytes) 8388608
//-size (bytes) 16777216
//-size (bytes) 33554432
-size (bytes) 134217728
//-size (bytes) 67108864
//-size (bytes) 1073741824

# Line size
//-block size (bytes) 8
-block size (bytes) 64

# To model Fully Associative cache, set associativity to zero
//-associativity 0
-associativity 1
//-associativity 4
//-associativity 8
//-associativity 16
```

FIGURE 5 – Paramètres de cache contenu dans le fichier cache.png

Initalement la taille du cache est de 134217728 bytes, soit 128 Mo. On peut retrouver les autres valeurs de la même manières, en particulier :

TABLE 7 – Paramètres de cache

Paramètre	**Valeur**
Taille de cache	128 Mo
Taille de bloc	64 octets
Associativité	1
Technologie	32 nm

4.2 Question 7

Notre version ne supportant pas le 28 nm, nous avons ciblé la technologie 32 nm. On peut déterminer la surface du cache L1 de A15 et 7 en modifiant les paramètres de *cache.cfg* tel que indiqué sur les Figures 3 et 4.

```
Cache Parameters:
    Total cache size (bytes): 32768
    Number of banks: 1
    Associativity: 2
    Block size (bytes): 32
    Read/write Ports: 1
    Read ports: 0
    Write ports: 0
    Technology size (nm): 32

    Access time (ns): 1.2689
    Cycle time (ns):  2.03685
    Precharge Delay (ns): 0
    Activate Energy (nJ): 0.00878296
    Read Energy (nJ): 0.0626449
    Write Energy (nJ): 0.0417095
    Precharge Energy (nJ): 0.0316549
    Leakage Power Closed Page (mW): 0.40658
    Leakage Power Open Page (mW): 0.704073
    Leakage Power I/O (mW): 0.697497
    Refresh power (mW): 6.92249e-05
    Cache height x width (mm): 0.265667 x 0.174962
```

FIGURE 6 – Paramètre du cache L1 pour le cortex **A7**

```
Cache Parameters:
    Total cache size (bytes): 32768
    Number of banks: 1
    Associativity: 2
    Block size (bytes): 64
    Read/write Ports: 1
    Read ports: 0
    Write ports: 0
    Technology size (nm): 32

    Access time (ns): 1.2689
    Cycle time (ns):  2.03685
    Precharge Delay (ns): 0
    Activate Energy (nJ): 0.00878296
    Read Energy (nJ): 0.0626449
    Write Energy (nJ): 0.0417095
    Precharge Energy (nJ): 0.0316549
    Leakage Power Closed Page (mW): 0.40658
    Leakage Power Open Page (mW): 0.704073
    Leakage Power I/O (mW): 0.697497
    Refresh power (mW): 6.92249e-05
    Cache height x width (mm): 0.265667 x 0.175632
```

FIGURE 7 – Paramètre du cache L1 pour le cortex **A15**

On sélectionne la donnée *'Cache height x width (mm)'*, ce qui nous donne une surface de $0.046\ mm^2$ pour le Cortex-A7 et $0.037\ mm^2$ pour le Cortex-A15. On en déduit le pourcentage de surface totale des coeurs :

Coeur	Pourcentage surface totale(%)	Taille coeur (hors Cache L1)(mm^2)
Cortex-A7	10	0.40
Cortex-A15	3	1.96

TABLE 8 – Paramètres de cache

Le cœur A15 avec sa taille de coeur bien supérieure s'adapte à des performances plus élevées, tandis que le cœur A7 est destiné à des tâches moins intensives en termes de calcul et en énergie. Le fait que les caches L1 représentent seulement 10% de la surface totale du Cortex A7 et 3% pour le Cortex A15 montre une optimisation de l'espace différente entre ces deux cœurs. Le Cortex A7, avec un pourcentage plus élevé, met davantage l'accent sur l'efficacité et la réduction de la latence à travers un accès rapide aux données et aux instructions. D'un autre côté, le Cortex A15, malgré un cache L1 plus grand en termes absolus, consacre un pourcentage plus faible de sa surface totale aux caches L1, indiquant que ce cœur pourrait avoir d'autres composants qui occupent plus d'espace (comme des unités d'exécution plus complexes, ou d'autres structures de cache plus grandes comme L2/L3) pour améliorer les performances globales. Le Cortex A7 est plus adapté aux systèmes où l'efficacité énergétique est cruciale, tandis que le Cortex A15 est préférable pour des applications nécessitant plus de puissance de calcul.

4.3 Question 8

On commence par calculer la surface occupée par les caches L2 en configurant le config.cfg de cette manière :

```
Cache Parameters:
    Total cache size (bytes): 524288
    Number of banks: 1
    Associativity: 8
    Block size (bytes): 32
    Read/write Ports: 1
    Read ports: 0
    Write ports: 0
    Technology size (nm): 32
```

FIGURE 8 – Paramètre du cache L2 pour le cortex **A7**

Coeur	Surface cache L2(mm^2)
Cortex-A7	0,48
Cortex-A15	0.42

TABLE 9 – Surface de Caches L2

La surface occupée par les caches L2 est de $0.31 mm^2$. On calcul donc la surface occupée par les caches L1 et faisant varier leur taille entre 1KB et 32KB comme à la question Q4 pour

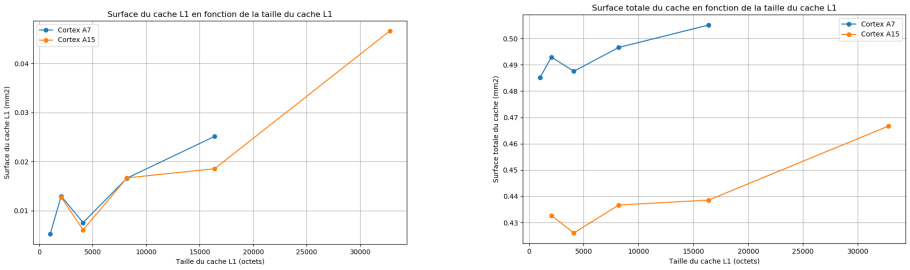
les deux Cortex, puis on ajoute la surface occupé par les Cortex et par les caches L2 (voir ***q8.py***) :

Taille L1	Surface L1 (mm^2)	Surface totale (mm^2)	IPC Dijkstra	IPC Blowfish
1024	0.005	0.485	1.420	0.203
2048	0.0129	0.492	1.550	0.207
4096	0.0075	0.487	1.610	0.216
8192	0.0166	0.496	1.640	0.224
16384	0.0251	0.505	1.680	0.233

TABLE 10 – Cortex-A7

Taille L1 (KB)	Surface L1 (mm^2)	Surface totale (mm^2)	IPC Dijkstra	IPC Blowfish
2048	0.013	0.432	1.810	0.258
4096	0.0060	0.426	1.860	0.266
8192	0.0066	0.436	1.880	0.275
16384	0.018	0.438	1.890	0.282
32768	0.046	0.466	2.030	0.284

TABLE 11 – Cortex-A15



(a) Surface caches L1

(b) Surface totale

FIGURE 9

4.4 Question 9

.

5 Efficacité énergétique

5.1 Question 10

On cherche la puissance consommée par chaque processeur à puissance maximale.

Puissance consommée = Consommation énergétique x Fréquence max

Taille L1(KB)	Efficacité surfacique Cortex-A7	Efficacité surfacique Cortex-A15
1024	2.93	/
2048	3.15	4.19
4096	3.31	4.37
8192	3.31	4.31
16384	3.33	4.32
32768	/	4.36

TABLE 12 – **Efficacité surfacique de chaque processeur pour le benchmark Dijkstra**

Taille L1	Efficacité surfacique Cortex-A7	Efficacité surfacique Cortex-A15
1024	0.42	/
2048	0.42	0.60
4096	0.44	0.62
8192	0.45	0.63
16384	0.46	0.64
32768	/	0.61

TABLE 13 – **Efficacité surfacique de chaque processeur pour le benchmark Blowfish**

Cortex	Cortex A16	Cortex A7
Consommation énergétique (mW/MHz)	0.10	0.20
Fréquence maximale (GHz)	1.0	2.5
Puissance consommée (mW)	100	500

TABLE 14 – Caractéristiques de deux processeurs en technologie 28nm

5.2 Question 11

A fréquence maximale, pour le cortex-A7, on divise les IPC trouvés TABLE 10 et TABLE 11 par 100mW. Pour le cortex-A15, on divise par 500mW. On obtient alors :

Taille L1	Efficacite energétique Cortex-A7	Efficacité energétique Cortex-A15
1024	0.01420	/
2048	0.01550	0.01810
4096	0.01610	0.01860
8192	0.01640	0.01880
16384	0.01680	0.01890
32768	/	0.02030

TABLE 15 – **Efficacité énergétique de chaque processeur pour le benchmark Dijkstra**

Taille L1	Efficacite energétique Cortex-A7	Efficacité energétique Cortex-A15
1024	0.000406	/
2048	0.000414	0.000516
4096	0.000432	0.000532
8192	0.000448	0.00055
16384	0.000466	0.000564
32768	/	0.000568

TABLE 16 – **Efficacité énergétique de chaque processeur pour le benchmark Blowfish**

6 Architecture système big.LITTLE

6.1 Question 12

L'efficacité énergétique d'un processeur mesure la quantité de travail accompli par le processeur par rapport à l'énergie qu'il consomme.Une efficacité énergétique plus élevée signifie que le processeur peut effectuer plus de travail tout en consommant moins d'énergie, ce qui est crucial pour les applications mobiles, les appareils alimentés par batterie et les centres de données soucieux de leur consommation d'énergie.

L'efficacité surfacique d'un processeur se réfère à la quantité de travail accompli par rapport à la surface physique occupée par le processeur. Une efficacité surfacique plus élevée signifie que le processeur peut effectuer davantage de travail sur une surface donnée, ce qui est crucial pour les appareils compacts tels que les smartphones, les tablettes et autres appareils embarqués où l'espace est limité.

6.1.1 Applications Dijkstra

Dans le cas de l'application de l'algorithme de Dijkstra pour trouver le plus court chemin dans un graphe, l'efficacité énergétique pourrait être plus privilégiéec elle rentre en jeu dans les applications embarquées ou mobiles où les ressources énergétiques sont limitées, comme les appareils IoT (Internet des objets), les systèmes autonomes... Une efficacité énergétique élevée permettrait à l'algorithme de fonctionner plus longtemps sur une batterie ou avec une consommation d'énergie réduite.
La configuration maximale semble être pour le cortex-A7 et pour n'importe quelle caractéristique considéré (surfacique ou énergétique), une taille L1 de 32KB, ce qui contredit les résultats trouvé en Q4. Ici on considère seulement l'IPC et non les autres paramètres considérés comme en Q4.

6.1.2 Applications Blowfish

Dans le cas de l'application de l'algorithme Blowfish pour le chiffrement de données, l'efficacité surfacique pourrait être plus pertinente. Le chiffrement de données est souvent effectué dans des environnements où l'espace physique est limité, comme les cartes à puce, les appareils mobiles ou les systèmes embarqués. Dans ces cas, l'efficacité surfacique devient cruciale car elle détermine la capacité à implémenter le chiffrement sur des puces électroniques avec des contraintes d'espace strictes, tout en maintenant des performances

acceptables.
De même ,la configuration optimale semble être pour le cortex-A7 et pour n'importe quelle caractéristique considéré (surfacique ou énergétique), une taille L1 de 32KB.