Laboratoire d'Annecy de Physique des Particules

What about Branching ?

Pierre Aubert

ESCAPE
cta cherenkov telescope array
cnrs
UNIVERSITÉ SAVOIE MONT BLANC

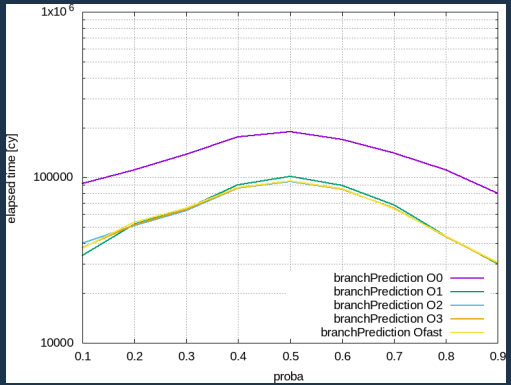
Kernel with branching

$$z_i = \begin{cases} x_i & \alpha_i < p \\ y_i & \alpha_i \geq p \end{cases}, \quad x_i, y_i \in \mathbb{R}, \quad p \in [0,1], \quad \alpha_i \in U(0,1), \quad i \in 1, N$$

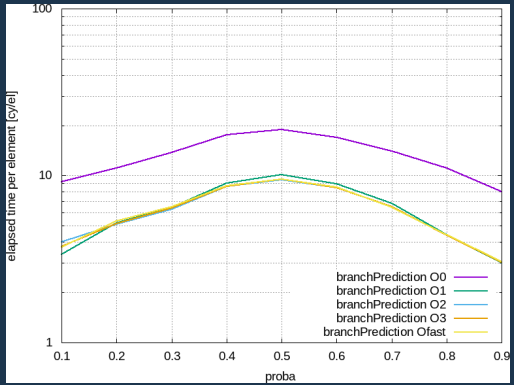
```
void dummyCopy(float* tabResult, const float * tabX, const float* tabY,
»       »       const float * tabProba, long unsigned int nbElement, float proba)
{
»       for(long unsigned int i(0lu); i < nbElement; ++i){
»       »       if(tabProba[i] < proba){
»       »       »       tabResult[i] = tabX[i];
»       »       }else{
»       »       »       tabResult[i] = tabY[i];
»       »       }
»       }
}
```

Branching probability : performances

Total Elapsed Time (cy)



Elapsed Time per element (cy/el)



Kernel with branching

```
void dummyCopy(float* tabResult, const float * tabX, const float* tabY,
»       »       const float * tabProba, long unsigned int nbElement, float proba)
{
»       for(long unsigned int i(0lu); i < nbElement; ++i){
»       »       if(tabProba[i] < proba){
»       »       »       tabResult[i] = tabX[i];
»       »       }else{
»       »       »       tabResult[i] = tabY[i];
»       »       }
»       }
}
```

Kernel with branching

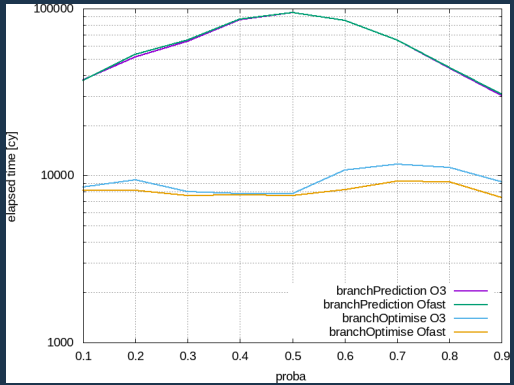
```
void dummyCopy(float* tabResult, const float * tabX, const float* tabY,
»       »       const float * tabProba, long unsigned int nbElement, float proba)
{
»       for(long unsigned int i(0lu); i < nbElement; ++i){
»       »       if(tabProba[i] < proba){
»       »       »       tabResult[i] = tabX[i];
»       »       }else{
»       »       »       tabResult[i] = tabY[i];
»       »       }
»       }
}
```

Change **if** into computing :

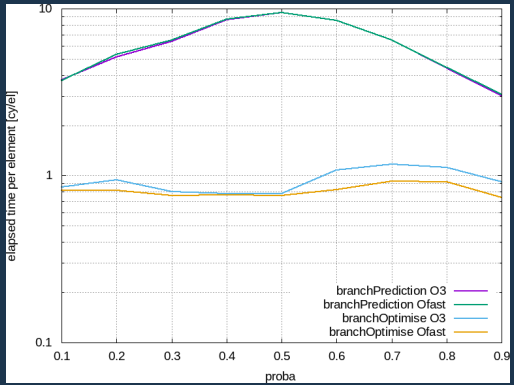
```
void dummyCopy(float* tabResult, const float * tabX, const float* tabY,
»       »       const float * tabProba, long unsigned int nbElement, float proba)
{
»       for(long unsigned int i(0lu); i < nbElement; ++i){
»       »       float cond(tabProba[i] < proba);
»       »       tabResult[i] = tabX[i]*cond + (1.0f - cond)*tabY[i];
»       }
}
```

Branching probability : optimisation

Total Elapsed Time (cy)



Elapsed Time per element (cy/el)



Computing Paradox

Version with **if** :

- **condition** : $\alpha_i < p$
- **if** : 3 cy
- $=$: equal operator

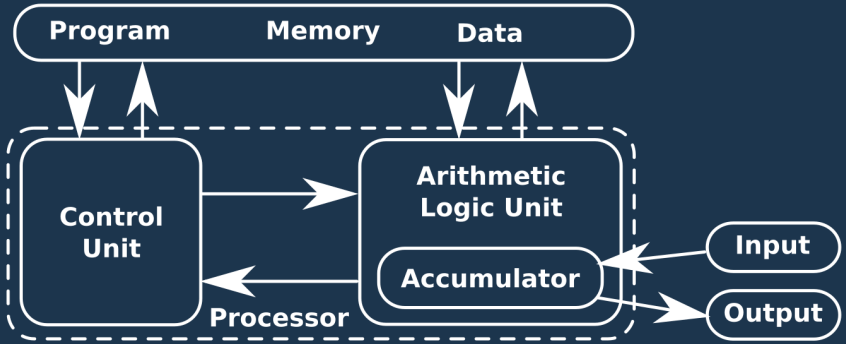
Specific computing : 3 cy

Version with computing :

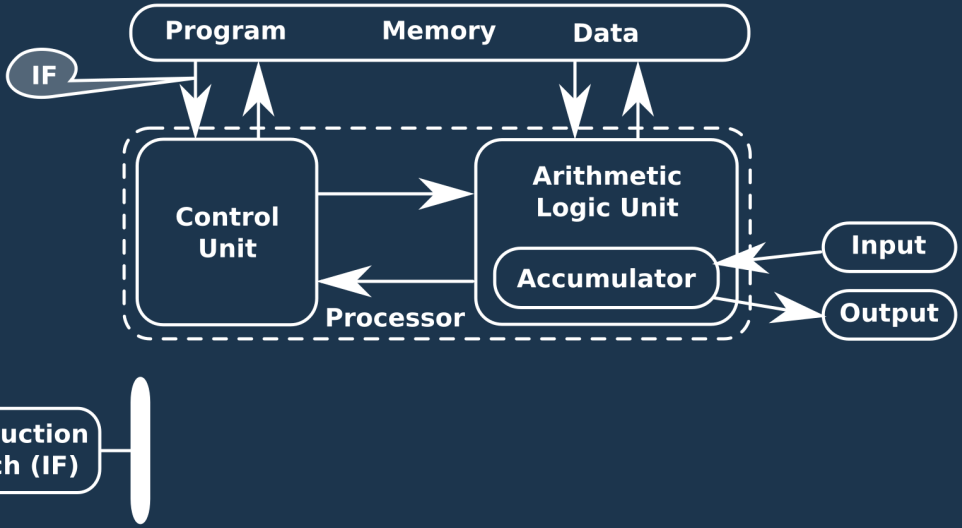
- **condition** : $\alpha_i < p$
- **2 multiplications** : 2×6 cy
- **addition + substraction** : 2×4 cy
- $=$: equal operator

Specific computing : 20 cy

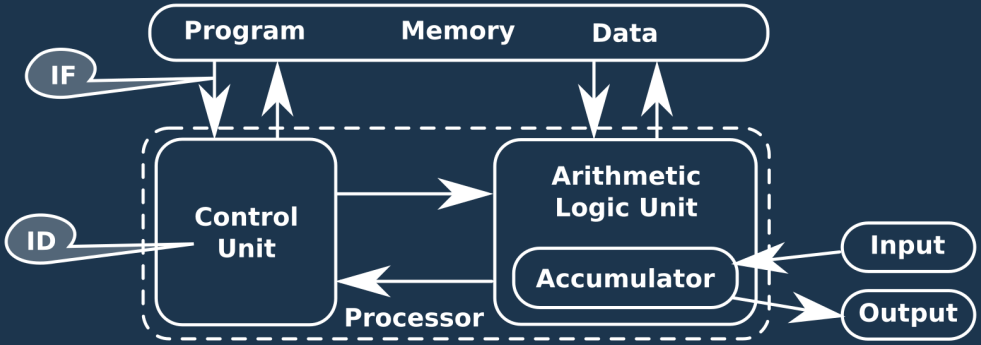
The CPU pipeline



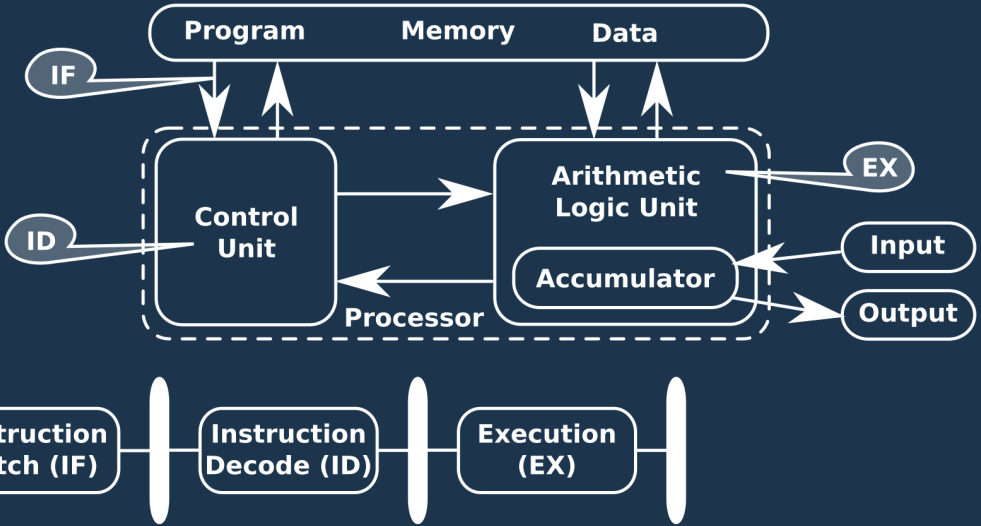
The CPU pipeline



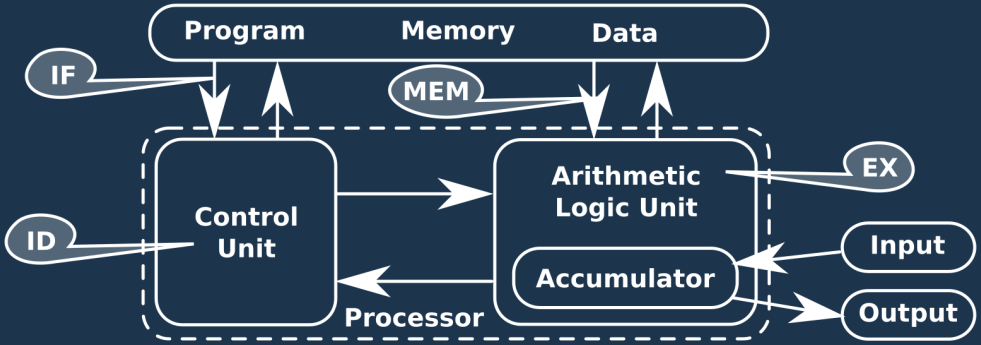
The CPU pipeline



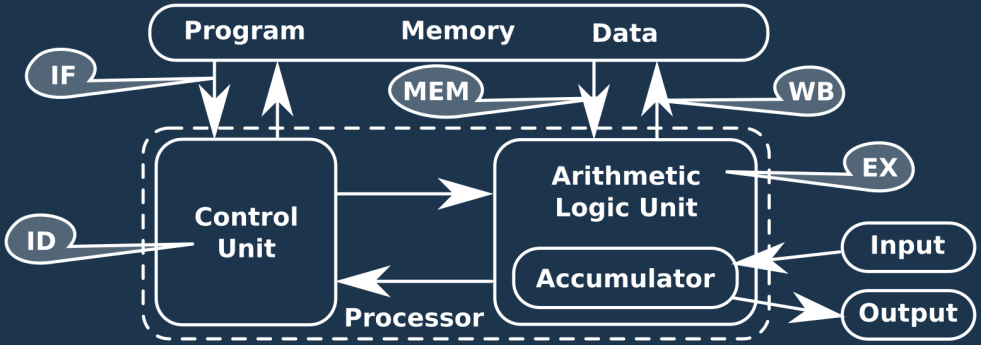
The CPU pipeline



The CPU pipeline

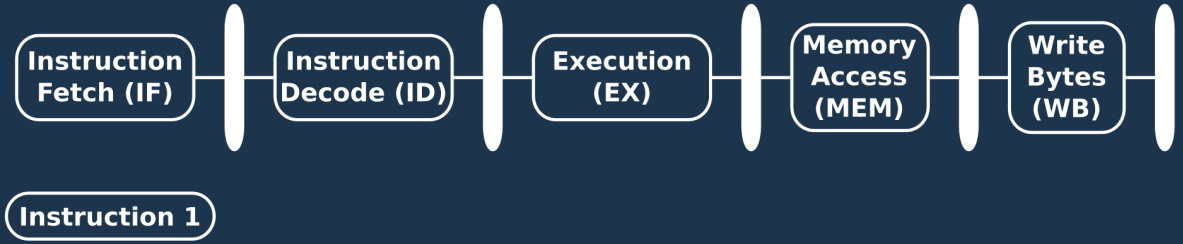


The CPU pipeline

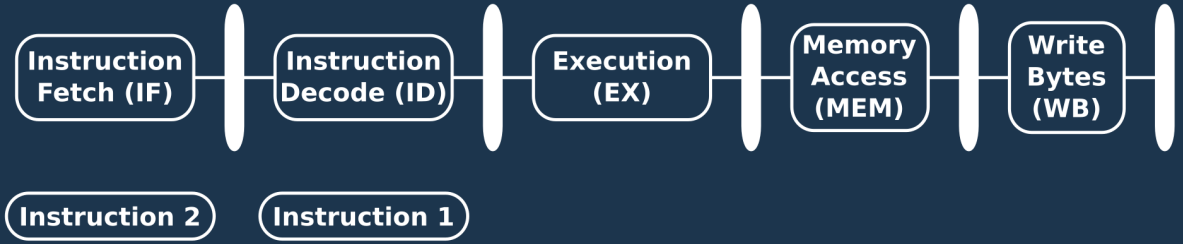


The CPU pipeline

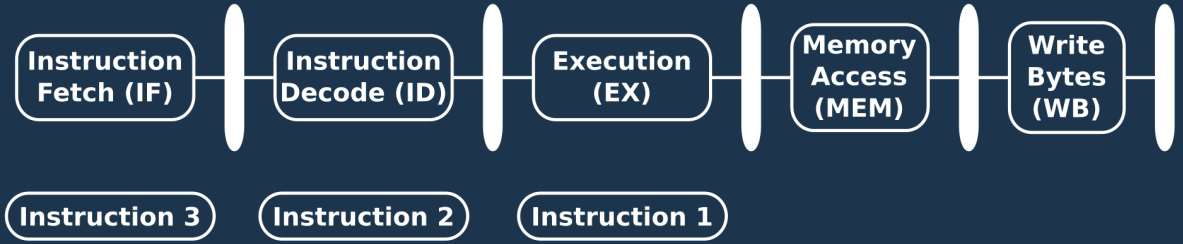
The CPU pipeline



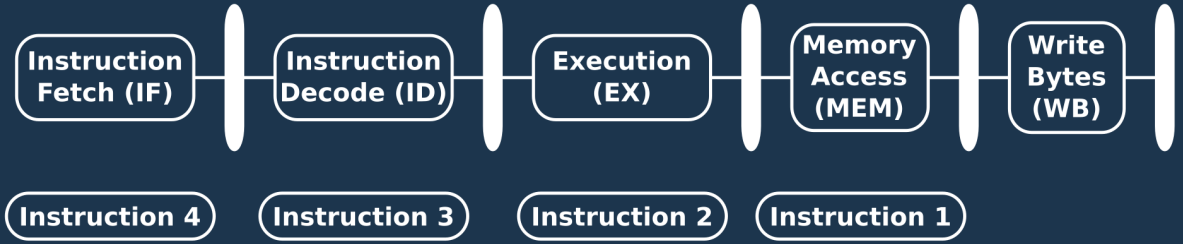
The CPU pipeline



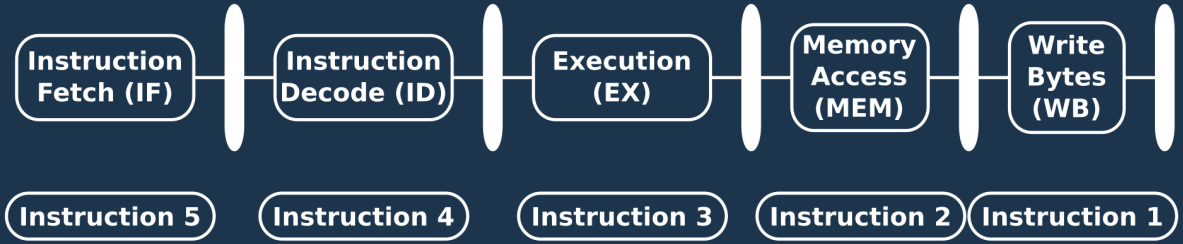
The CPU pipeline



The CPU pipeline



The CPU pipeline

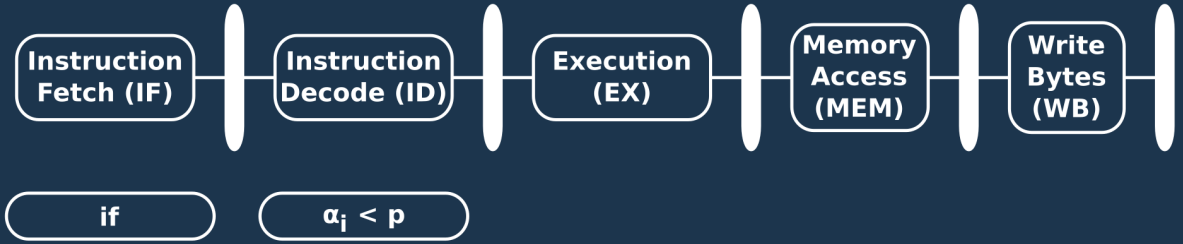


The CPU pipeline with branching

The CPU pipeline with branching

$\alpha_i < p$

The CPU pipeline with branching



The CPU pipeline with branching

$z_i = x_i$

$z_i = y_i$

?

if

$\alpha_i < p$

The CPU pipeline with branching

$z_i = x_i$

$z_i = y_i$

?

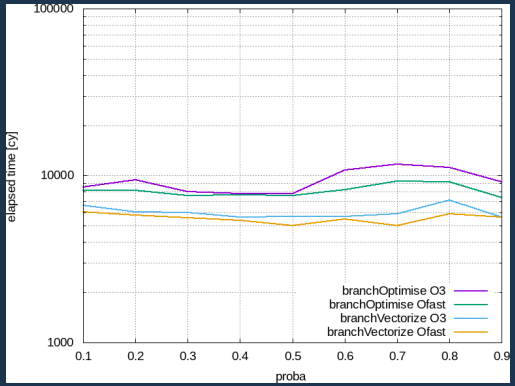
if

$\alpha_i < p$

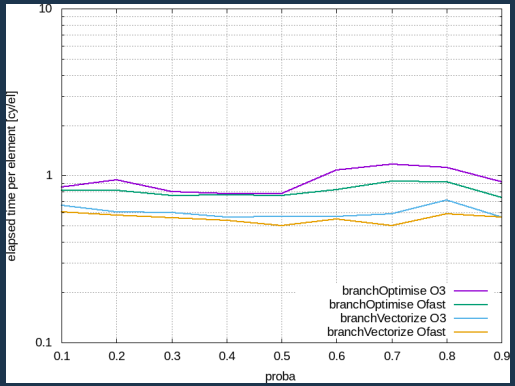
Branching Prediator

Branching probability : vectorize

Total Elapsed Time (cy)

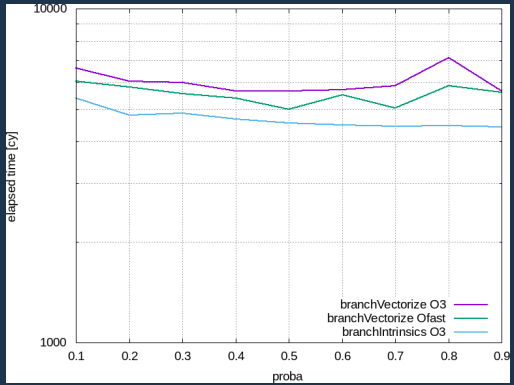


Elapsed Time per element (cy/el)

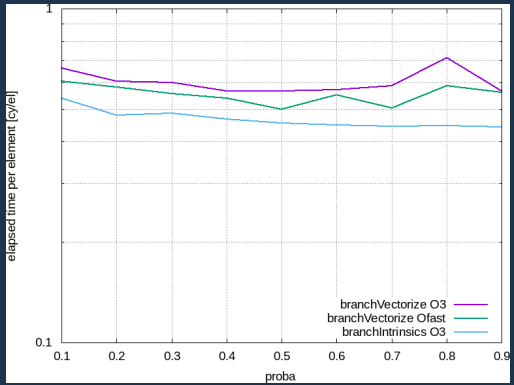


Branching probability : intrisics

Total Elapsed Time (cy)

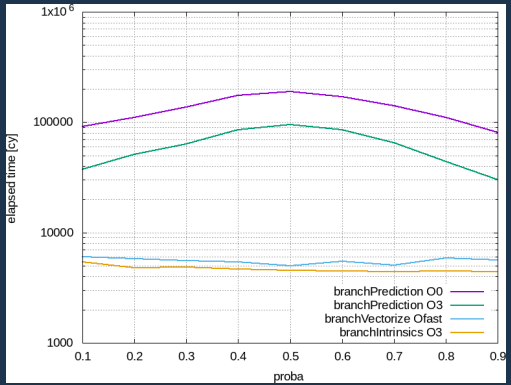


Elapsed Time per element (cy/el)



Branching probability : summary

Total Elapsed Time (cy)



Elapsed Time per element (cy/el)