ENSICAEN
Computer Science

FAMILLE SANDY BRIDGE

Architecture et Technologie des Ordinateurs

Hugo Descoubes - Novembre 2013

Analysons maintenant les principaux mécanismes d'accélération matérielle apportés au fil des années et rencontrées sur architecture Sandy Bridge de Intel*. Une bonne partie de ces mécanismes d'optimisation n'ont pas forcément à être connus des développeurs, même pour un développeur bas niveaux.*

Contrairement à d'autres architectures (par exemple, CPU DSP C6xxx de TI), les CPU's compatibles x86-64 exécutent un flot d'instruction présent in-order en mémoire et appliquent des mécanismes d'accélération matérielle d'exécution out-of order dans le CPU. Ceci amènent notamment une architecture hardware plus complexe, plus gourmande en énergie, sujette à de fortes contraintes d'échauffement, plus difficile à appréhender et à accélérer.

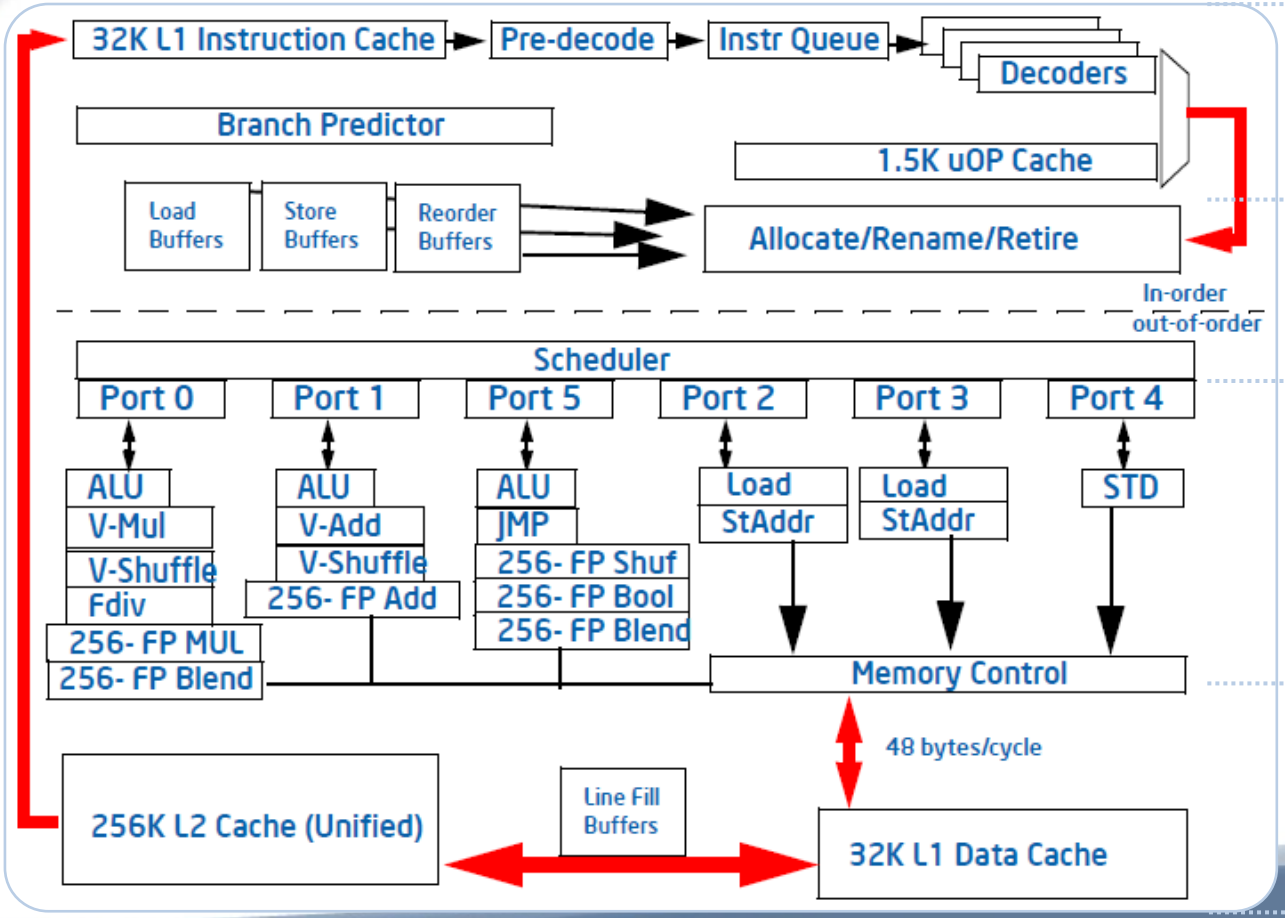
Observons les principales évolutions des architectures x86-64 du 8086 jusqu'à l'architecture Sandy Bridge :

CPU Architecture	Année	Espace d'adressage Linéaire/physique	Principales évolutions
8086	**1978**	16bits (logical)/20bits (phys.)	Jeu d'instruction x86 original, segmentation mémoire
80186	**1982**		Accélération matérielle (calcul d'adresses, multiplication ,division …)
80286		16bits (logical)/30bits (linear)/24bits (phys.)	MMU (Memory Managment Unit) avec segmented mode, tables GDT (Global et Local Descriptor Table) et LDT, modes protégés avec privilèges
80386	**1985**	32bits (logical)/46bits (linear)/32bits (phys.)	Jeu d'instruction 32bits (IA-32), MMU avec unité de Pagination
80486	**1989**		Pipeline RISC-like, mémoire cache intégrée et FPU (Floating Point Unit)
Pentium	**1993**		Processeur superscalaire (exécution plusieurs instructions par cycle CPU via plusieurs unités d'exécution), extension MMX
Pentium pro (P6)	**1995**	32bits (logical)/46bits (linear)/36bits (phys.) **via PAE**	Cache L2 intégré, PAE 4bits (Physical Address Extension), translation micro-instructions (uop), exécution out-of order, exécution spéculative (register renaming)
Pentium II Pentium III	**1997**		Cache L3 intégré, extension SSE (instructions SIMD)
Pentium 4	**2000**		Extension SSE2 , Hyper-Threading, pipeline profond, uop cache (Trace Cache)

Observons les principales évolutions des architectures x86-64 du 8086 jusqu'à l'architecture Sandy Bridge :

CPU Architecture	Année	Espace d'adressage Linéaire/physique	Principales évolutions
Pentium 4 Prescott	**2004**	*Cf. CPUID*	*Jeu d'instruction x64 (Intel 64), extension SSE3*
Core 2	**2006**		*Multi-cœurs, extension SSE4 (Penryn), faible consommation*
Atom	**2008**	*Cf. CPUID* *Sur ma machine Intel 64 famille Sandy Bridge sous Linux 48bits (linear-virtual)/36bits (phys.)*	*Très faible consommation, exécution in-order (marchés laptop et mobile)*
Corei7 Nehalem	**2008**		*3 niveaux de cache intégrés, bus QPI (remplaçant FSB vers chipet), extension pour cryptage AES*
Sandy Bridge Ivy Bridge	**2011**		*Extension SSE5 et AVX, GPU, advanced uop cache*

Vu d'ensemble du pipeline matériel de la microarchitecture Sandy Bridge de Intel :



http://www.intel.com

L1 ICache
32Ko
8ways set associative

ITLB
4Ko page 128 entry
2-4Mo page 8 entry

Etudions le pipeline matériel de la microarchitecture Sandy Bridge de Intel sortie courant 2011.

- *La mémoire cache et la notion de TLB (Translation Lookaside Buffer) seront étudiés plus tard dans le cours. Nous partirons de l'hypothèse (souvent juste) que les instructions nécessaire à l'exécution du code en cours traitement par le CPU sont présentent en ICache (Instruction Cache).*

L1 ICache
32Ko
8ways set associative

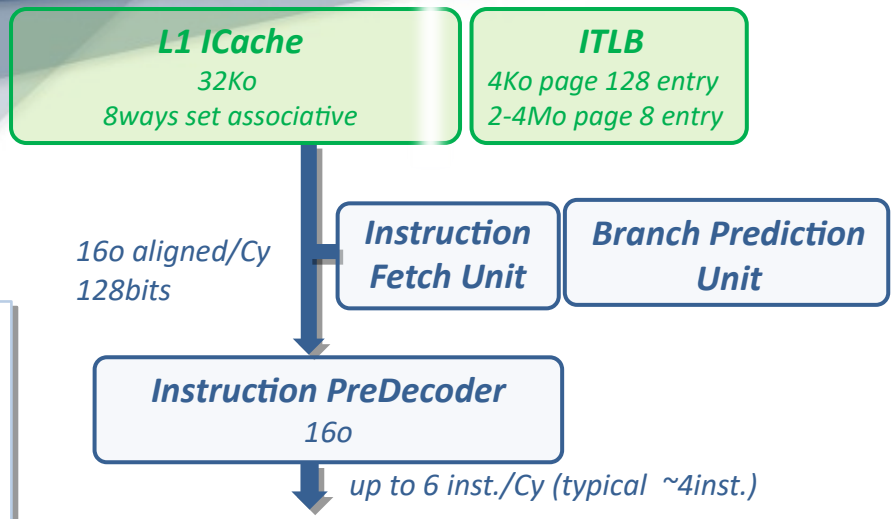
ITLB
4Ko page 128 entry
2-4Mo page 8 entry

Branch Prediction Unit

Legacy Decode Pipeline (Nehalem)

- *L'unité de prédiction aux branchements est chargée d'anticiper et prédire les branchements futurs. Elle peut prédire de façon efficace : branchement conditionnel, jump et call directs et indirects, return de procédure.*

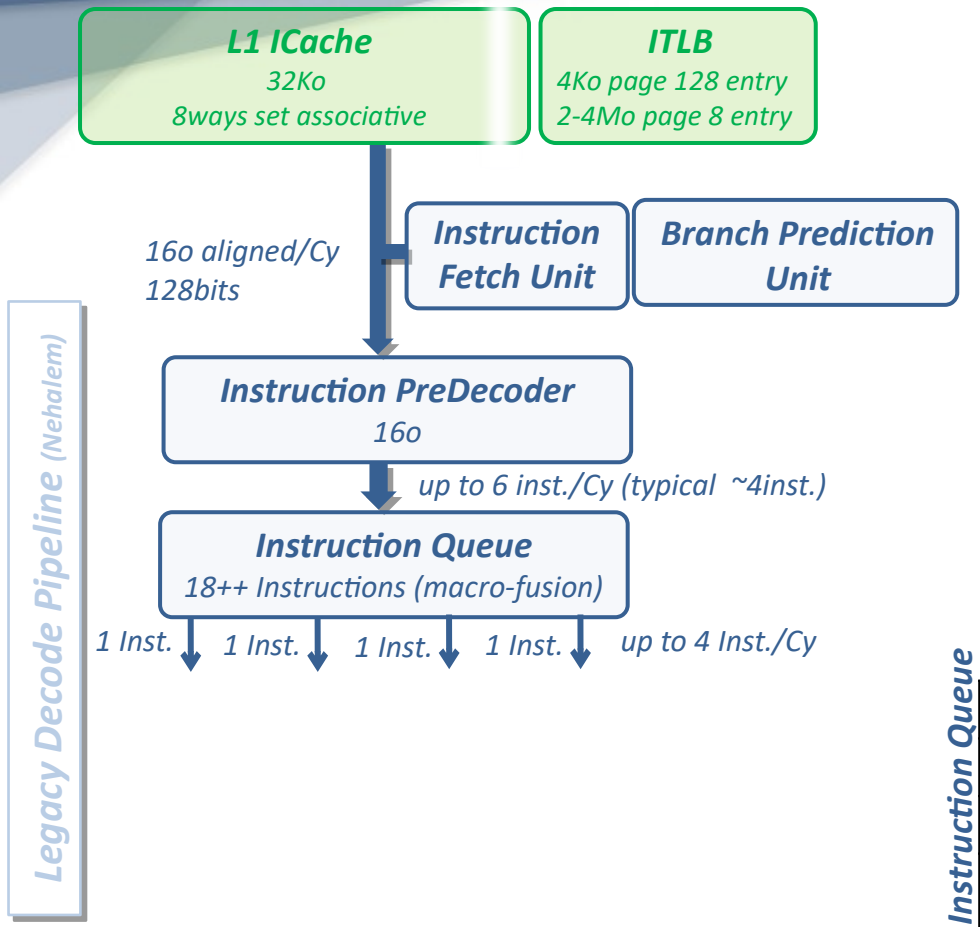
- *Le pipeline actuellement présenté est hérité de la microarchitecture Nehalem de Intel (premiers corei7 fin 2008).*



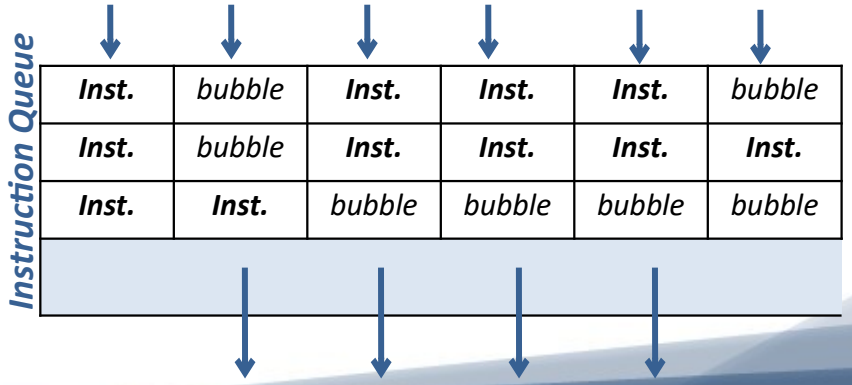
Legacy Decode Pipeline (Nehalem)

- *L'unité Instruction Fetch est chargée d'acheminer 16octets alignés présents en L1 Icache vers l'unité de pré-décodage.*

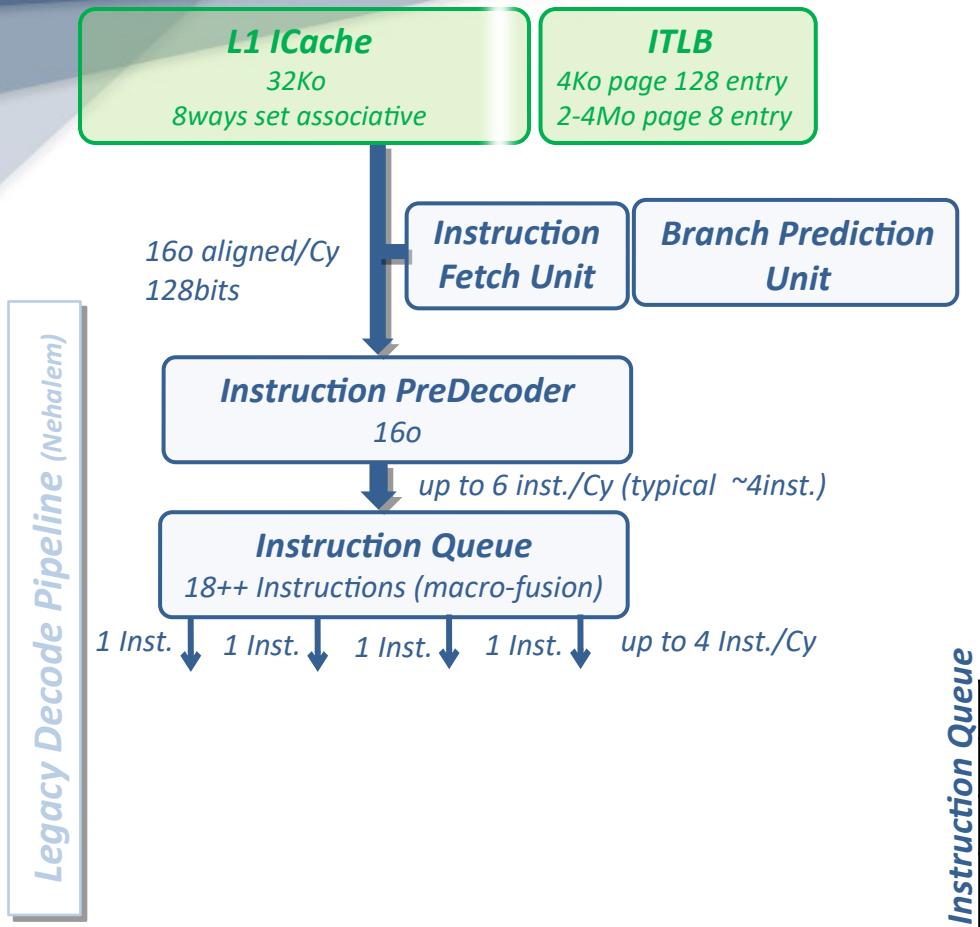
- *Comme tout jeu d'instruction CISC, les instructions ne possèdent pas une taille fixe. L'unité de pré-décodage est chargée d'identifier le nombre d'instruction contenue dans les 16o récupérés (instructions préfixées).*



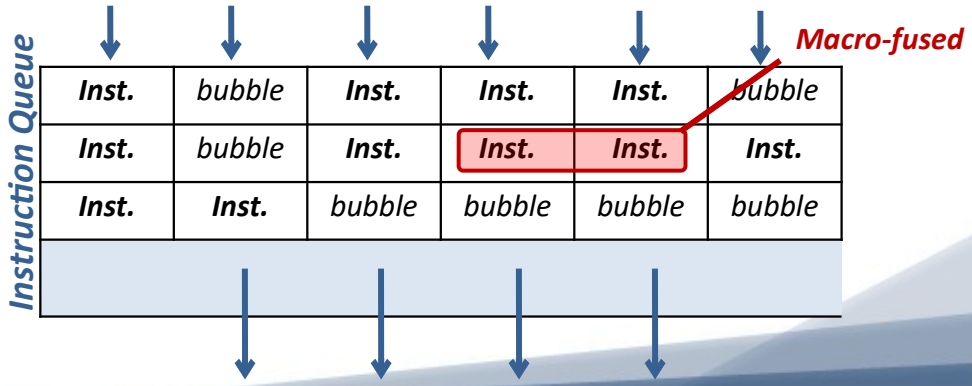
- *Potentiellement, des ''bulles'' (clusters dans la file d'attente sans instruction) peuvent être naturellement insérés en fonction du code en cours d'exécution.*



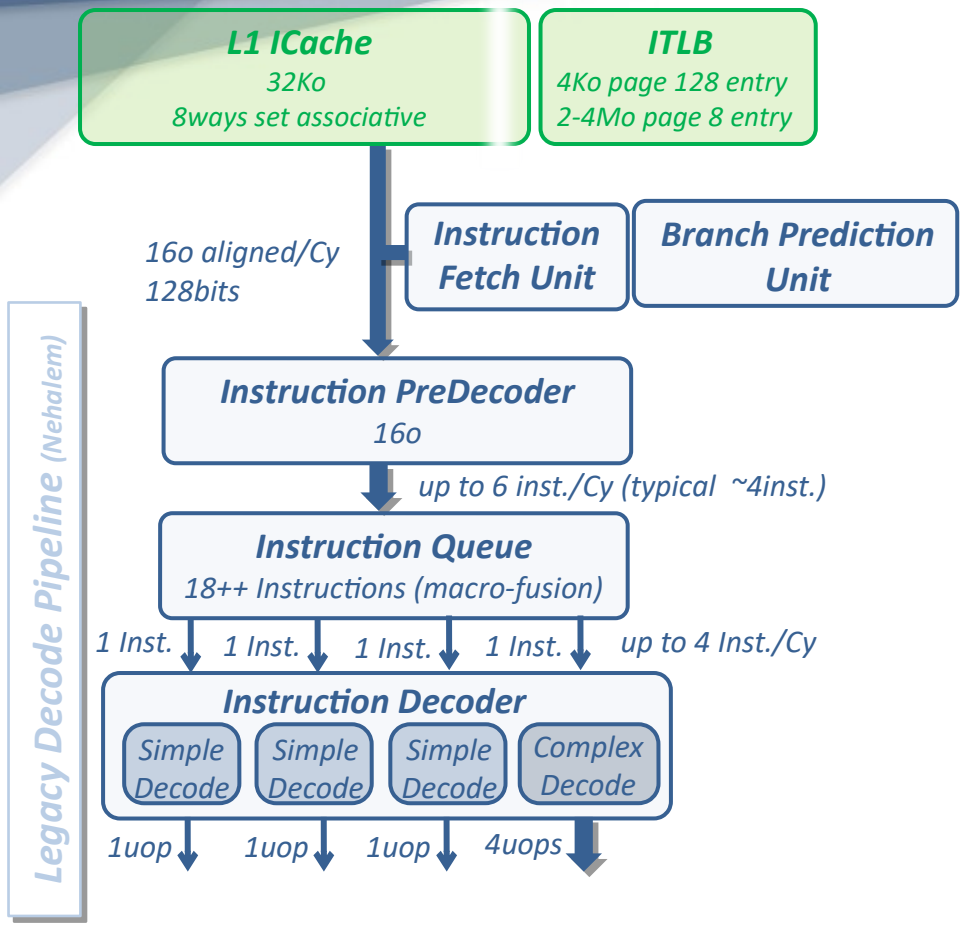
Front End – Out-of Order Engine – Execution Core – Famille Haswell



- *Cet étage peut également réaliser la macro-fusion d'instruction simple. Sous condition, prenons des exemples d'instructions pouvant fusionnées sur architecture Sandy Bridge : CMP, TEST, ADD, SUB, AND, INC, DEC*



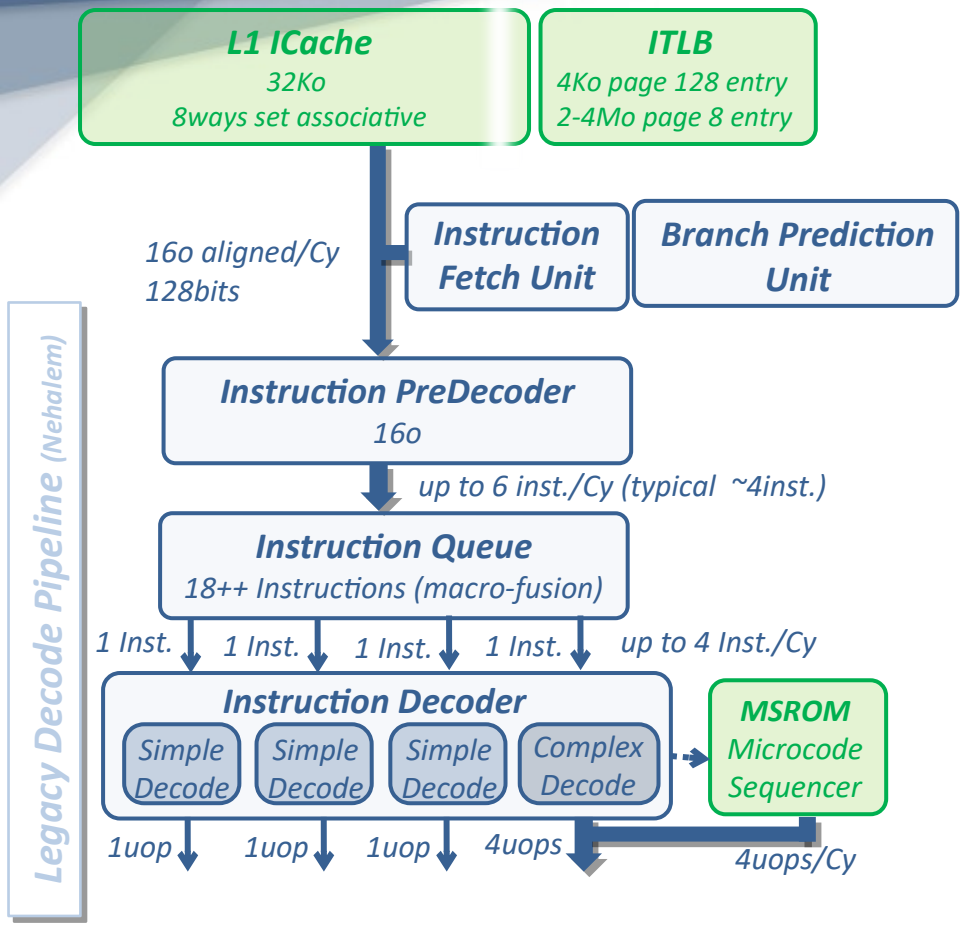
Front End – Out-of Order Engine – Execution Core – Famille Haswell



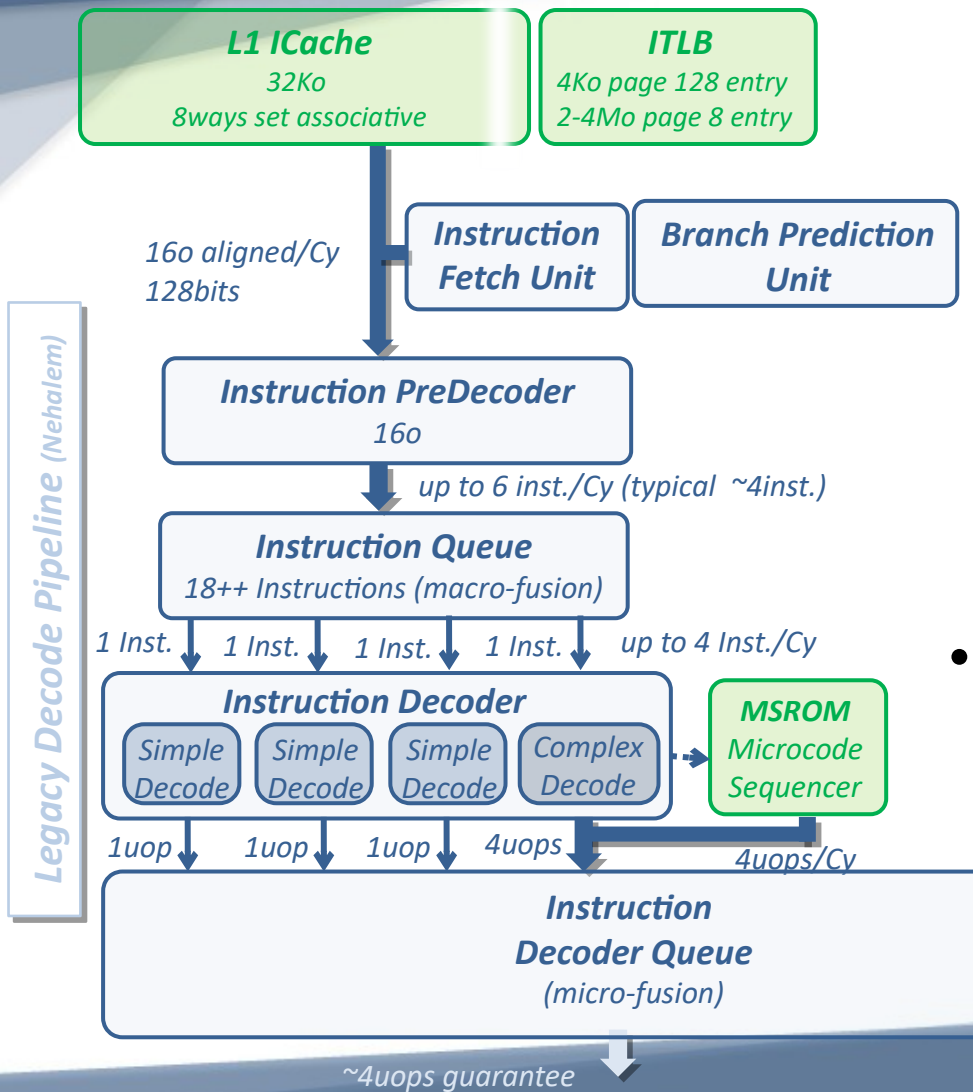
- *L'unité de décodage d'instruction est chargée de décoder les instructions CISC en micro-opérations ou uops élémentaires (operation, load, store, jump …). Par exemple,* ***xorq %rax, 32(%rbp)*** *devient* ***xorOperation+memoryStore***

- *Par exemple, lorsque Intel introduisit l'extension SSE2 avec le Pentium4, chaque instruction sur 128bits était découpée en 2 uops de 64bits.*

Front End – Out-of Order Engine – Execution Core – Famille Haswell



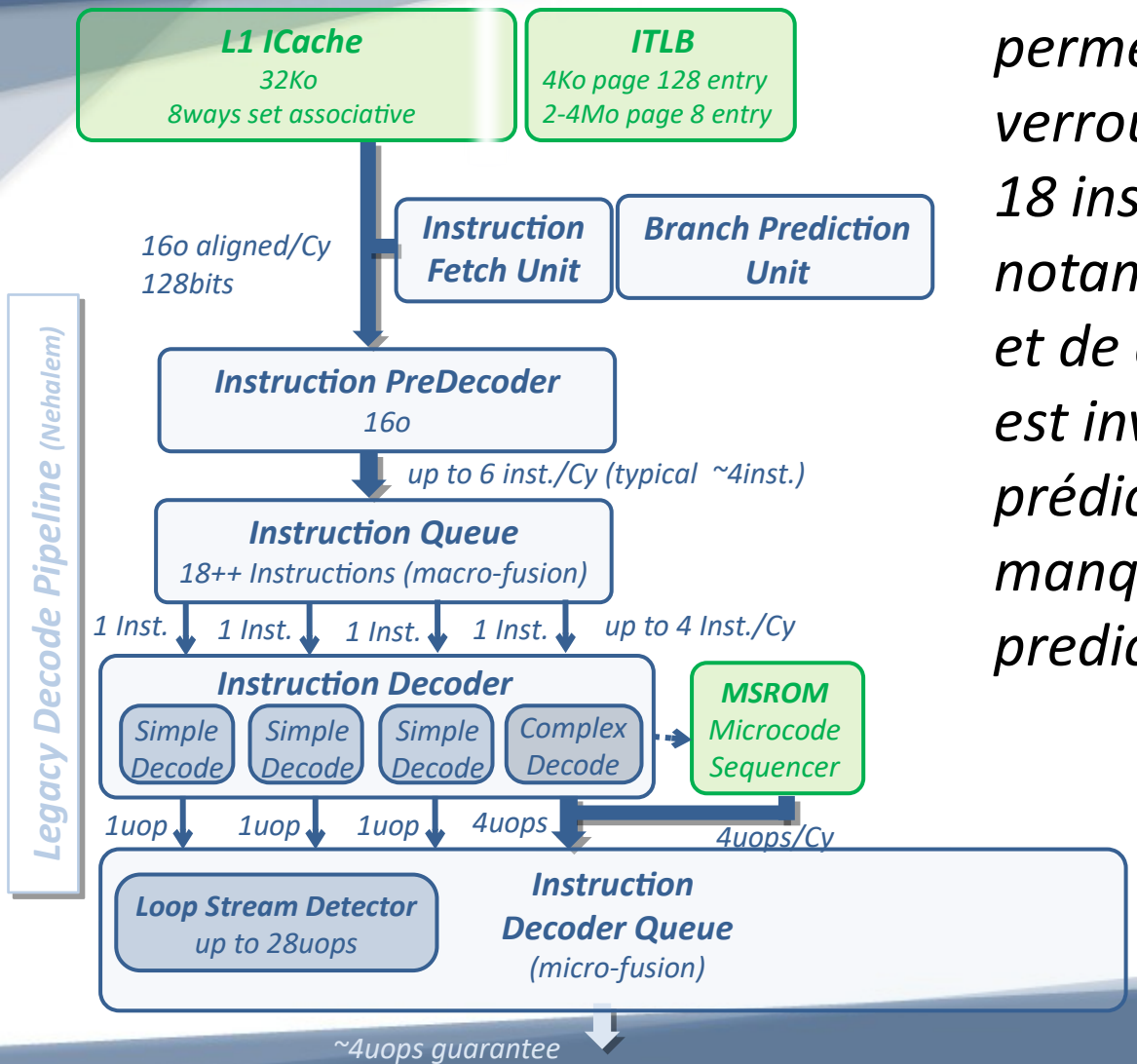
- *Les instructions CISC complexe sont routées vers un décodeur dédié (Complex Decode). En cas d'instructions générant plus de 4 uops, celles-ci sont alors extraites d'une mémoire morte ROM associée à un séquenceur (MSROM) chargé d'envoyer le flot de uops à l'étage suivant.*



- *Le flot d'instruction est alors acheminé vers une nouvelle file d'attente dont le travail principal est de combler les ''bulles'' rencontrées dans les étages précédents afin d'assurer un flot dans le désordre quasi constant de 4uops (out-of order).*

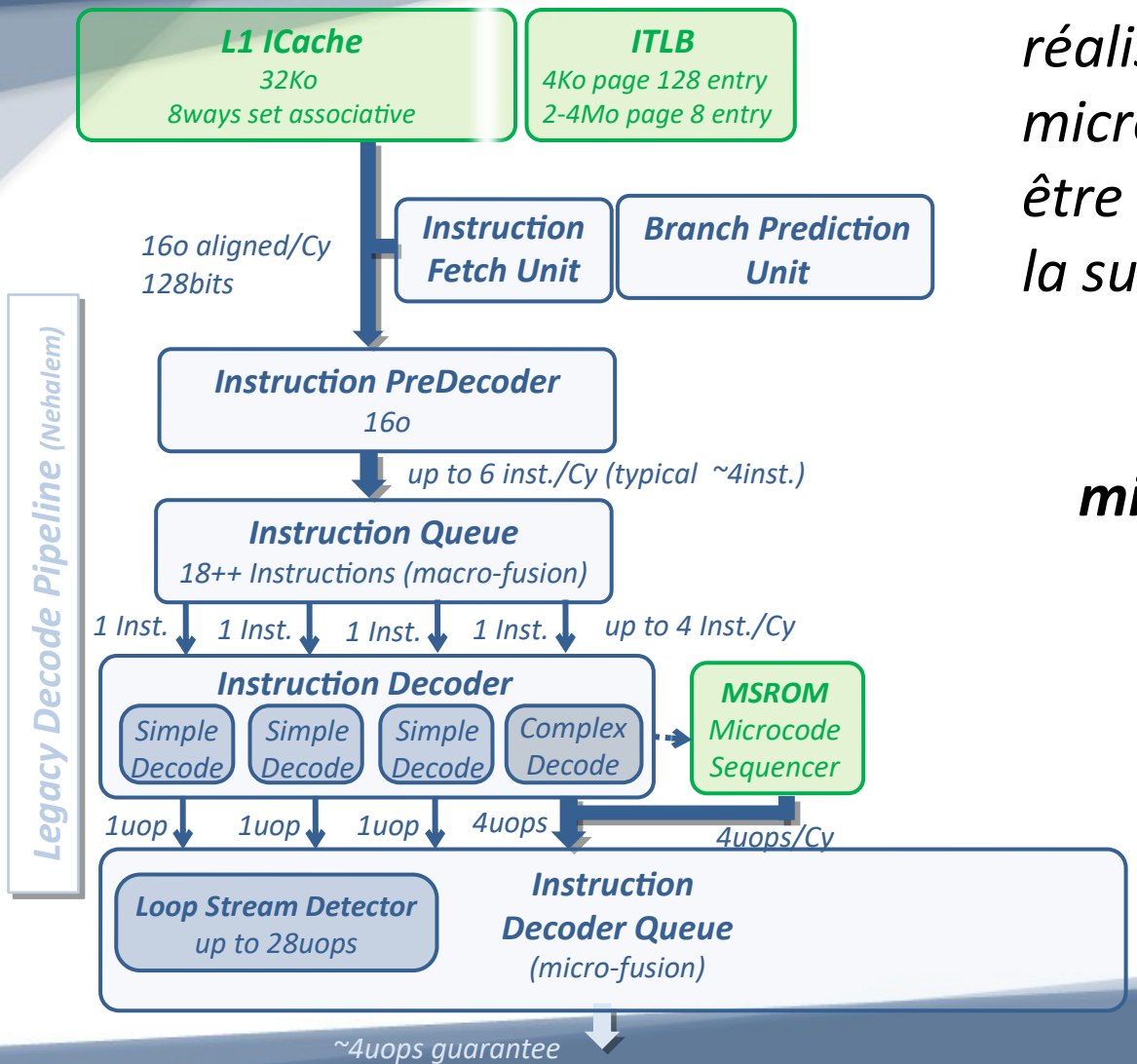
- *Certains mécanismes d'accélération sont insérés à cette étape. Etudions les principaux.*

Front End – Out-of Order Engine – Execution Core – Famille Haswell



- *Le Loop Stream Detector (LSD) permet de détecter et de verrouiller les petites boucles (< 18 instructions) et évitent donc notamment des phases de fetch et de décodage inutiles. La boucle est invalidée par l'unité de prédiction dès qu'une prédiction manquée est détectée (miss-prediction).*

Front End – Out-of Order Engine – Execution Core – Famille Haswell



- *Cet étage peu également réaliser sous condition des micro-fusions de uops pouvant être décodées et exécutées par la suite :*

uop + uop

=

micro-fused complex uop

Observons l'instruction dpps précédemment étudiée dans le cours et découvrons son découpage en uops (unfused, micro-fused et macro-fused) à avant l'étage d'exécution out-of order du CPU :

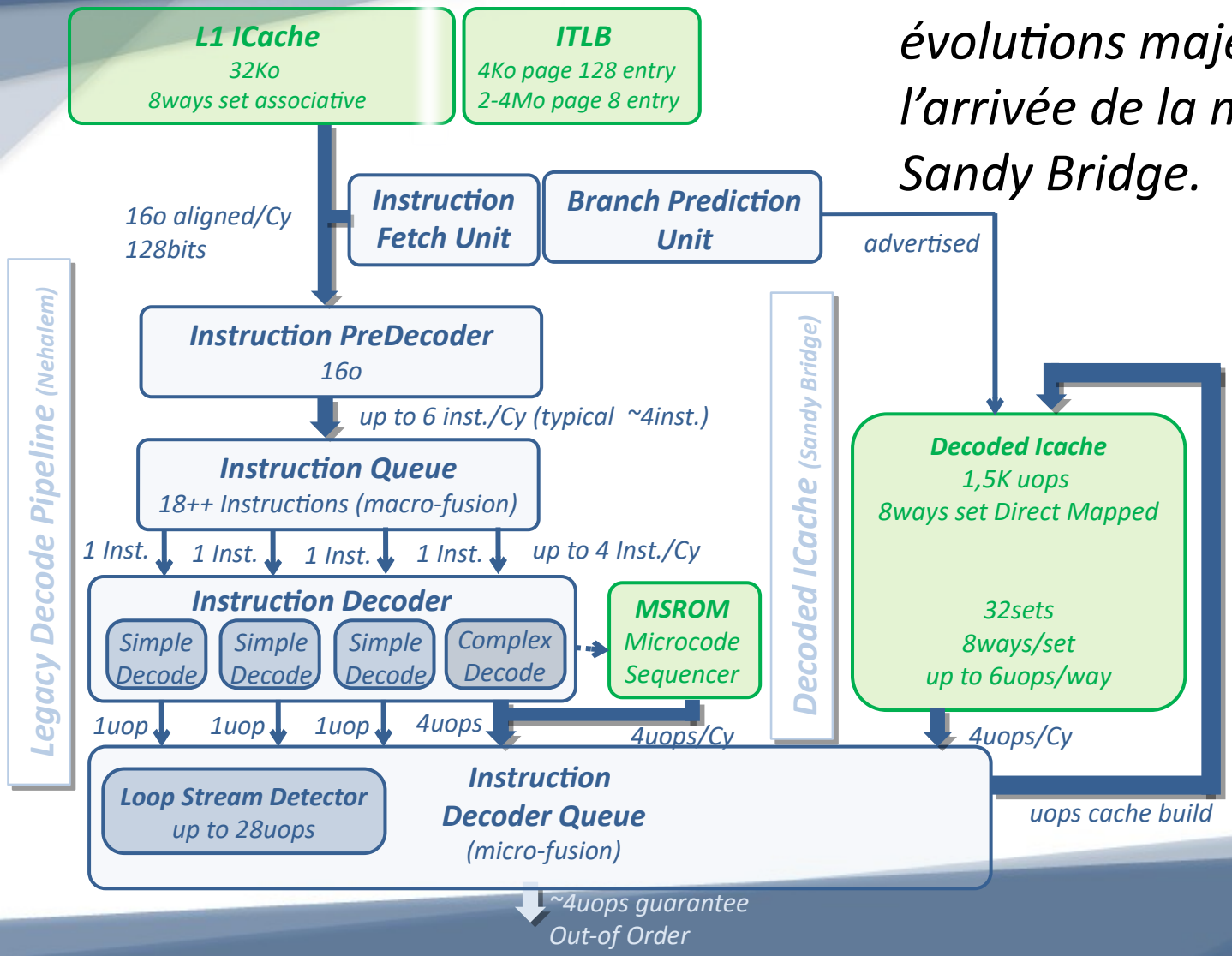
DPPS — Dot Product of Packed Single Precision Floating-Point Values

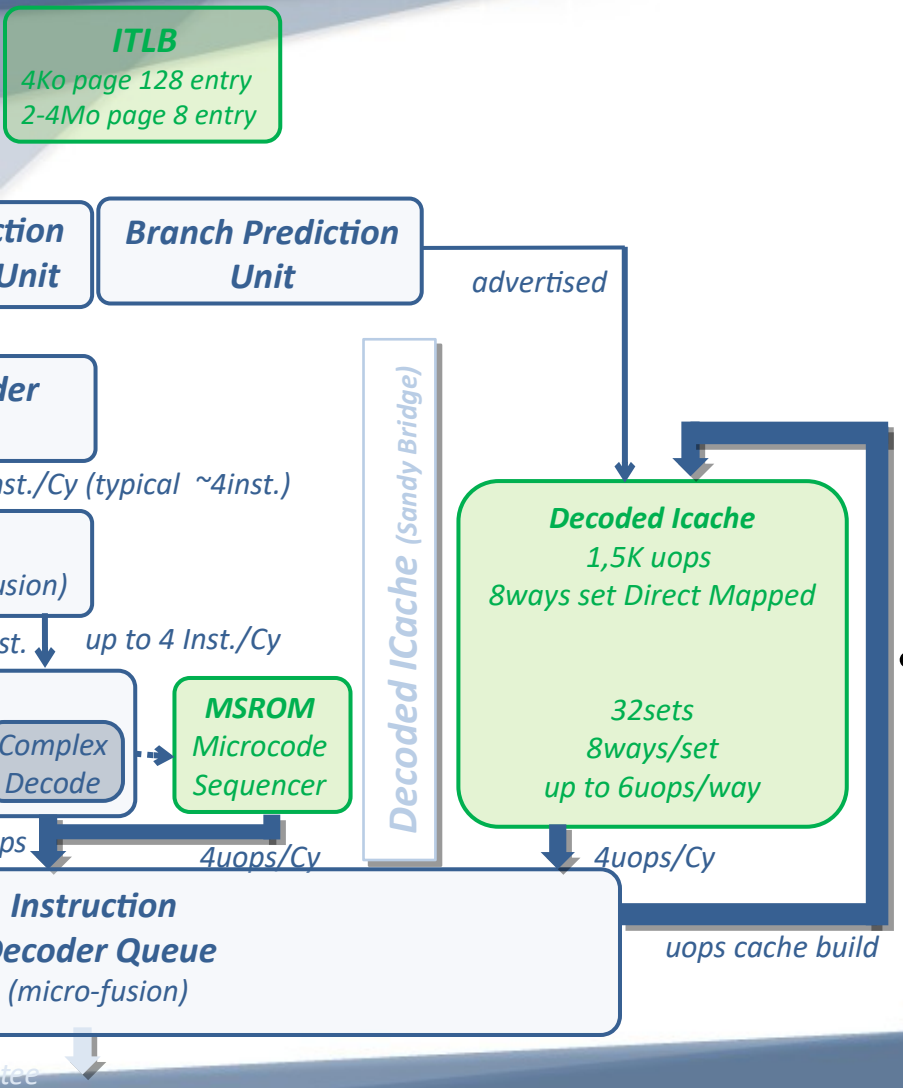
Opcode/ Instruction	Op/ En	64/32-bit Mode	CPUID Feature Flag	Description
66 0F 3A 40 /r ib DPPS *xmm1, xmm2/m128, imm8*	RMI	V/V	SSE4_1	Selectively multiply packed SP floating-point values from *xmm1* with packed SP floating-point values from *xmm2*, add and selectively store the packed SP floating-point values or zero values to *xmm1*.

Instruction	*Operands*	*uops fused*	*unfused uops*					*Latency*	*Reciprocal Throughput*	*Comments*
			p0	*p1*	*p5*	*p23*	*p4*			
dpps	*xmm, xmm, imm8*	*4*								*SSE4.1*
	xmm, m128, imm8	*6*								

Front End – Out-of Order Engine – Execution Core – Famille Haswell

- *Le Decoded Icache est l'une des évolutions majeure amenée avec l'arrivée de la micro-architecture Sandy Bridge.*





- ***Le Decoded Icache est une mémoire cache de uops de taille fixe*** *déjà décodées et donc remplie après l'étage de décodage d'instruction. Cette mémoire cache est enfouie dans l'architecture du cœur. Il s'agit d'une technologie proche du Trace Cache du Pentium 4.*

- *Il faut savoir que sur un flow typique d'instructions, ~80% des uops sont issues de ce cache, ~15% du pipeline hérité et ~5% du MSROM.*

Il existe plusieurs familles d'architectures capables d'exécuter un flot d'instructions dans le désordre (out-of order). *Les architectures Intel (depuis le Pentium pro famille P6), les PowerPC de IBM, les Cortex-A de ARM … effectuent ces mécanismes d'accélération dans le CPU au niveau de l'étage d'exécution (après décodage).*

TEXAS INSTRUMENTS

D'autres architectures comme les processeurs spécialisés DSP C6000 de Texas Instruments effectuent ces mécanismes à la compilation. L'avantages étant d'avoir une architecture CPU beaucoup plus simple mais un code out-of order en mémoire contrairement aux architectures précédemment citées qui possèdent un code in-order en mémoire et donc plus facile à lire et à debugger pour le développeur.

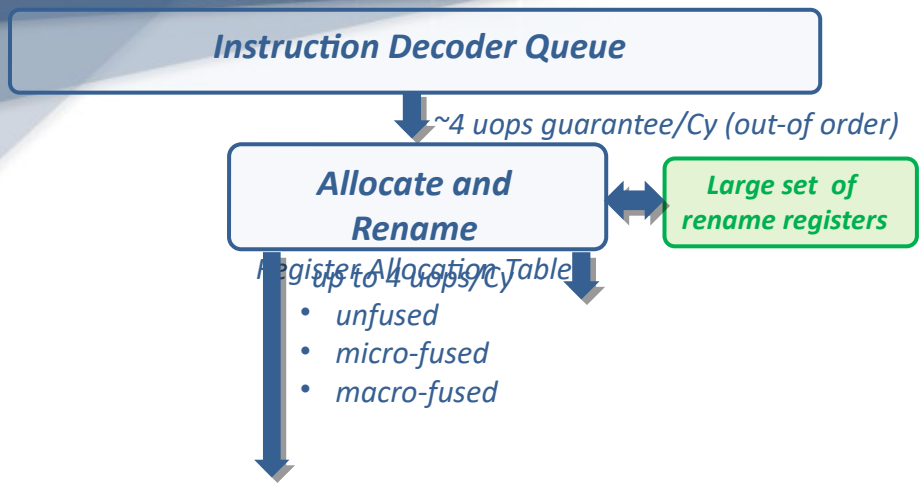
Prenons un exemple de code en assembleur C6000 de TI. Observons ce même code non optimisé (in-order) et légèrement optimisé (out-of order). Architecture superscalaire non-exploitée :

Memory program in order

```
; void function ( 16, 1 );        ;function call 10cy
MOVKL       16, A4  ;delayslot 0cy, parameter n°1
MOVKL       1, B4   ;delayslot 0cy, parameter n°2
MOVKL       retAdd, B3      ;delayslot 0cy, save return address
MOVKH       retAdd, B3      ;delayslot 0cy, save return address
B       function        ;delayslot 5cy, call function
NOP     5
; jump to function and return here after!
```

Memory program out-of order

```
; void function ( 16, 1 );        ;function call 6cy
B       function        ;delayslot 5cy, call function
MOVKL       16, A4  ;delayslot 0cy, parameter n°1
MOVKL       1, B4   ;delayslot 0cy, parameter n°2
MOVKL       retAdd, B3      ;delayslot 0cy, save return address
MOVKH       retAdd, B3      ;delayslot 0cy, save return address
NOP     1
; jump to function and return here after!
```



Etudions l'étage d'exécution out-of order de micro-opérations (exécution spéculative) :

- *L'étage d'allocation et de renommage est chargé d'allouer des ressources au flot de uops (registres temporaires renommés). Cette unité travail avec un très large jeu de registres non accessibles au développeur et enfouis dans l'architecture du CPU (registres entiers 160 entry, flottant 144 entry…)*

Rappelons l'exécution et le découpage en micro traitements de l'instruction dpps précédemment rencontrée :

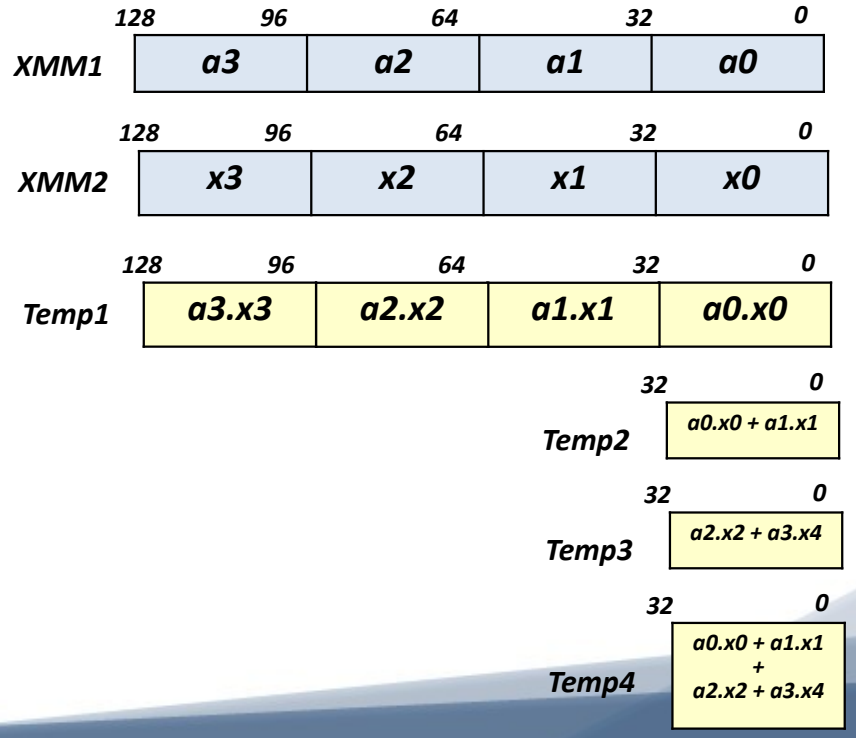
dpps 0xF1, %xmm2,%xmm1

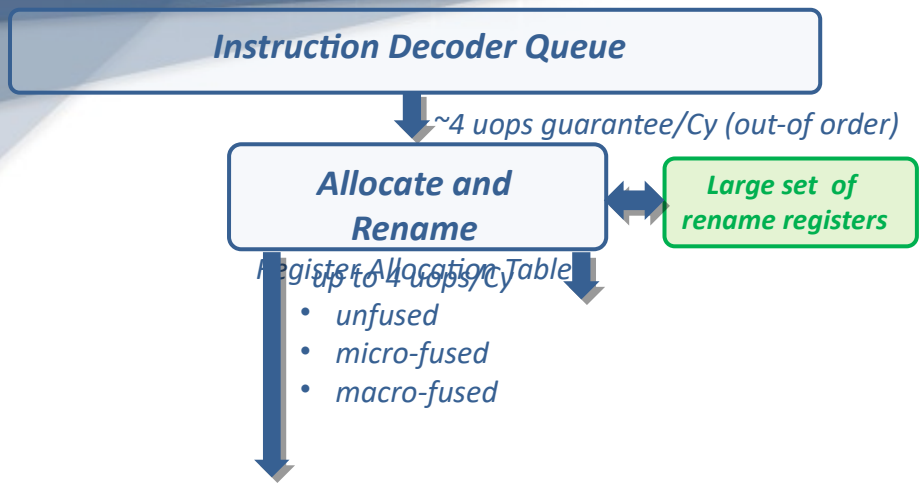
Operation

```
DP_primitive (SRC1, SRC2)
IF (imm8[4] = 1)
    THEN Temp1[31:0] ← DEST[31:0] * SRC[31:0];
    ELSE Temp1[31:0] ← +0.0; FI;
IF (imm8[5] = 1)
    THEN Temp1[63:32] ← DEST[63:32] * SRC[63:32];
    ELSE Temp1[63:32] ← +0.0; FI;
IF (imm8[6] = 1)
    THEN Temp1[95:64] ← DEST[95:64] * SRC[95:64];
    ELSE Temp1[95:64] ← +0.0; FI;
IF (imm8[7] = 1)
    THEN Temp1[127:96] ← DEST[127:96] * SRC[127:96];
    ELSE Temp1[127:96] ← +0.0; FI;

Temp2[31:0] ← Temp1[31:0] + Temp1[63:32];
Temp3[31:0] ← Temp1[95:64] + Temp1[127:96];
Temp4[31:0] ← Temp2[31:0] + Temp3[31:0];
```

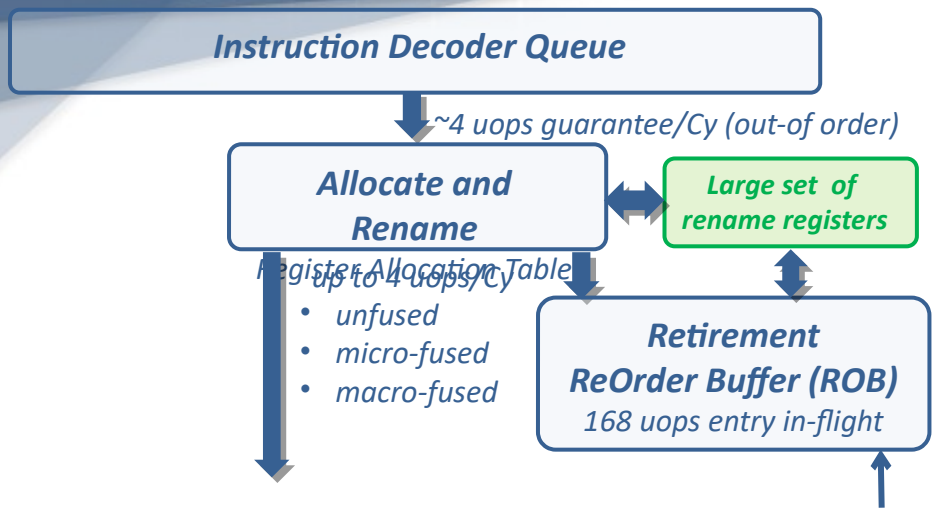
XMMi (i = 0 à 15 with Intel 64)
128bits General Purpose Registers
for SIMD Execution Units



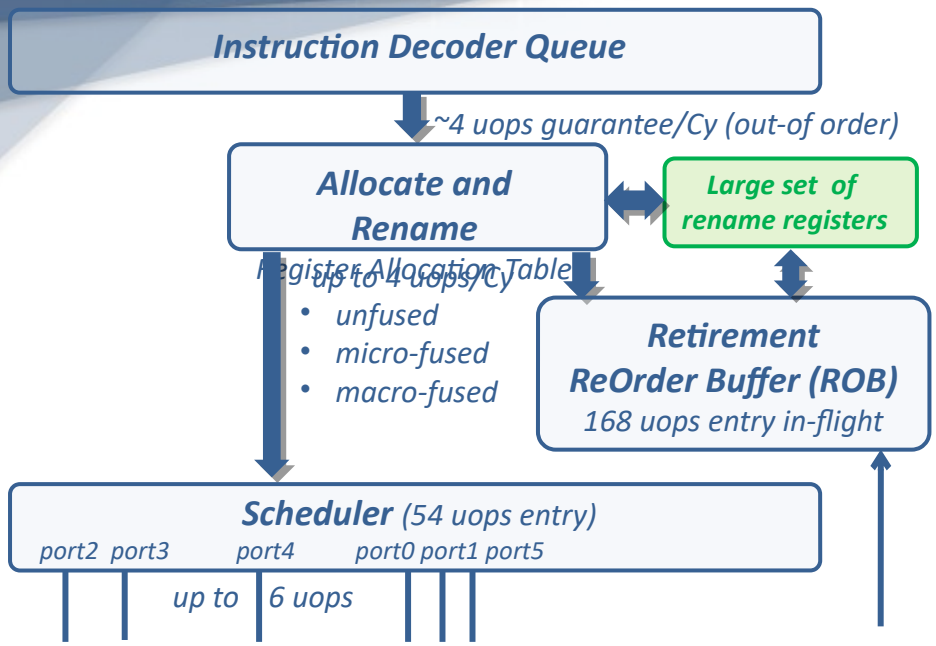


- *L'étage d'allocation et de renommage peut également exécuter puis retirer du pipeline certaines instructions simples dîtes idiomatiques :*

✓ ***nop***
✓ ***Zero idioms*** *(sub reg,reg, xor reg,reg …)*
✓ ***…***



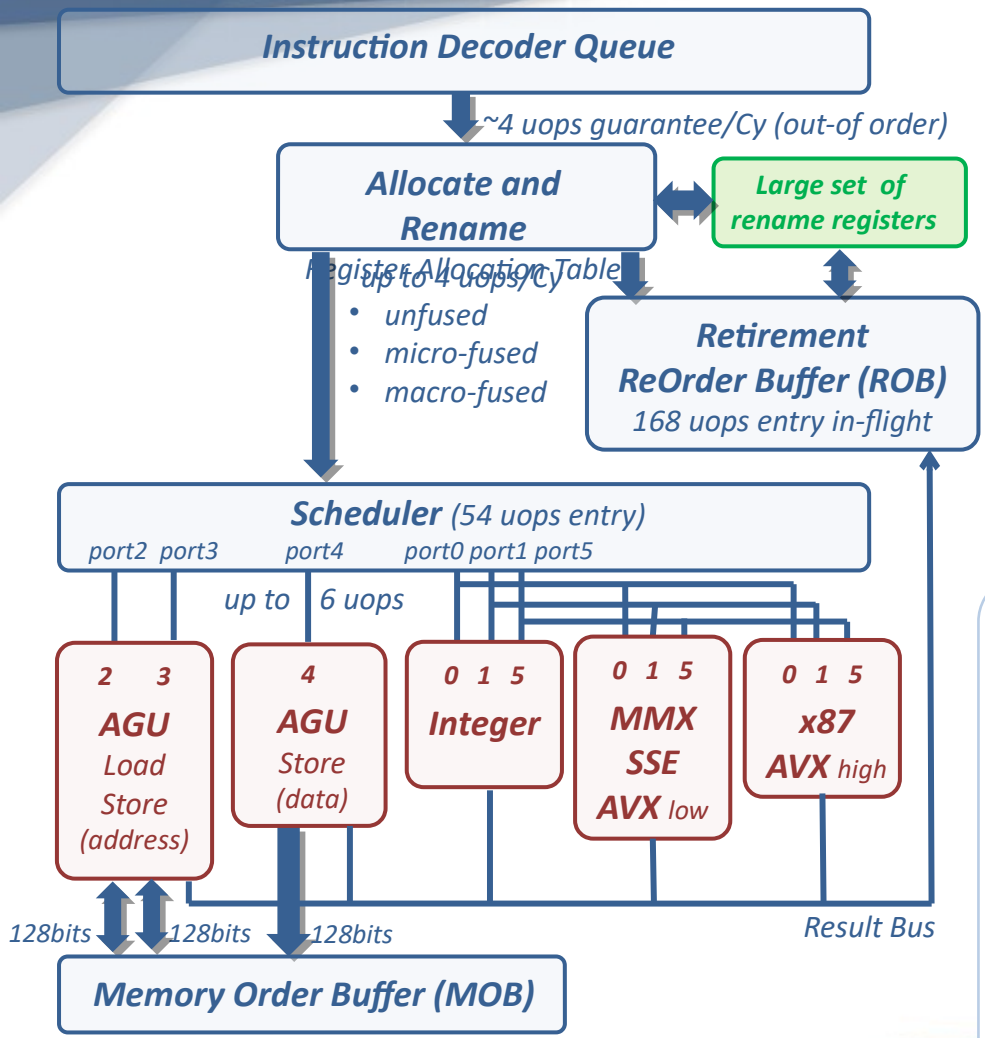
- *L'étage de retirement peut supporter jusqu'à 168 uops en vol. Cet étage est chargé de retirer les uops de la file d'attente après s'être assurer du bon résultat suite au passage dans l'étage d'exécution. L'unité de retirement est également chargée de capturer les défauts et exceptions matérielles (stop l'exécution des uops à la source des défauts pour le processus courant).*



Etudions l'étage d'exécution :

- *Le scheduler ou ordonnanceur charge dans une fille d'attente (jusqu'à 54 uops) les uops prêtes à être exécutées jusqu'à ce que les opérandes sources soient également prêtes. Le scheduler réparti alors les uops vers les unités d'exécution correspondantes (jusqu'à 6uops/cycle).*

Front End – Out-of Order Engine – Execution Core – Famille Haswell



- *L'étage d'exécutions possède une architecture dîtes **superscalaire**. Exécution de plusieurs uops en parallèle sur un même cycle CPU via plusieurs unités d'exécutions. Observons les familles de uops supportés par chaque port :*

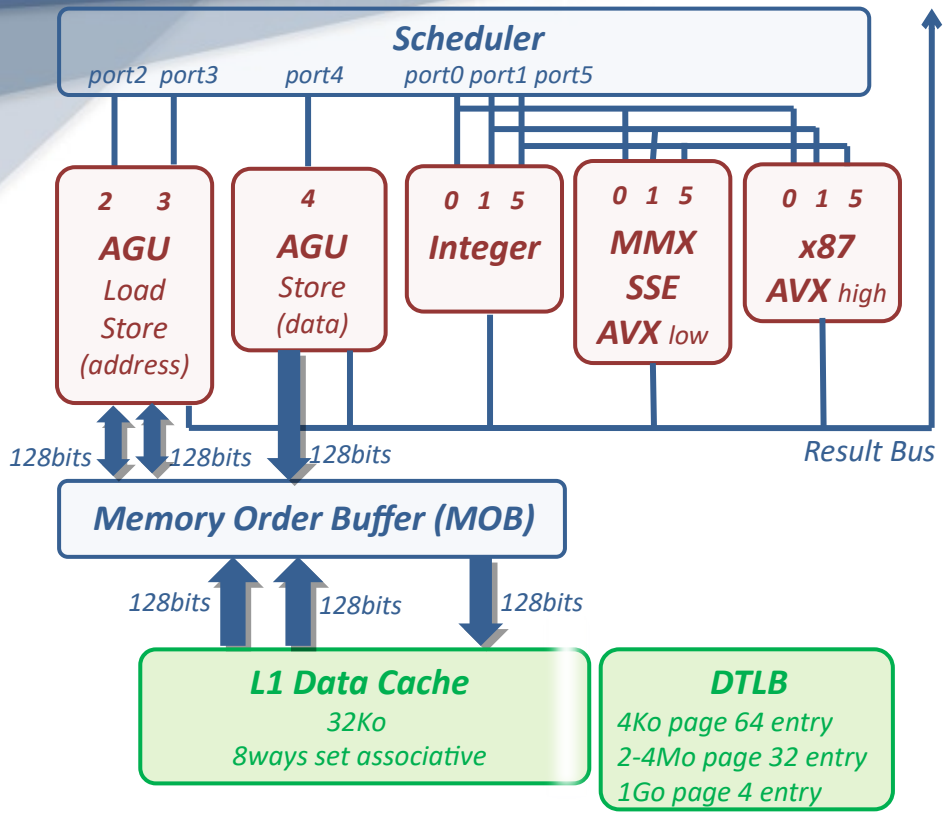
	Port 0	Port 1	Port 2	Port 3	Port 4	Port 5
Integer	ALU, Shift	ALU, Fast LEA, Slow LEA, MUL	Load_Addr, Store_addr	Load_Addr Store_addr	Store_data	ALU, Shift, Branch, Fast LEA
SSE-Int, AVX-Int, MMX	Mul, Shift, STTNI, Int-Div, 128b-Mov	ALU, Shuf, Blend, 128b-Mov			Store_data	ALU, Shuf, Shift, Blend, 128b-Mov
SSE-FP, AVX-FP_low	Mul, Div, Blend, 256b-Mov	Add, CVT			Store_data	Shuf, Blend, 256b-Mov
X87, AVX-FP_High	Mul, Div, Blend, 256b-Mov	Add, CVT			Store_data	Shuf, Blend, 256b-Mov

Poursuivons l'étude de l'instruction dpps et de son passage dans le pipeline matériel d'une architecture Sandy Bridge :

DPPS — Dot Product of Packed Single Precision Floating-Point Values

Opcode/ Instruction	Op/ En	64/32-bit Mode	CPUID Feature Flag	Description
66 0F 3A 40 /r ib DPPS *xmm1, xmm2/m128, imm8*	RMI	V/V	SSE4_1	Selectively multiply packed SP floating-point values from *xmm1* with packed SP floating-point values from *xmm2*, add and selectively store the packed SP floating-point values or zero values to *xmm1*.

Instruction	*Operands*	*uops fused*	*unfused uops*					*Latency*	*Reciprocal Throughput*	*Comments*
			p0	*p1*	*p5*	*p23*	*p4*			
dpps	*xmm, xmm, imm8*	*4*	*1*	*2*	*1*	*-*	*-*	*12*	*2*	*SSE4.1*
	xmm, m128, imm8	*6*								



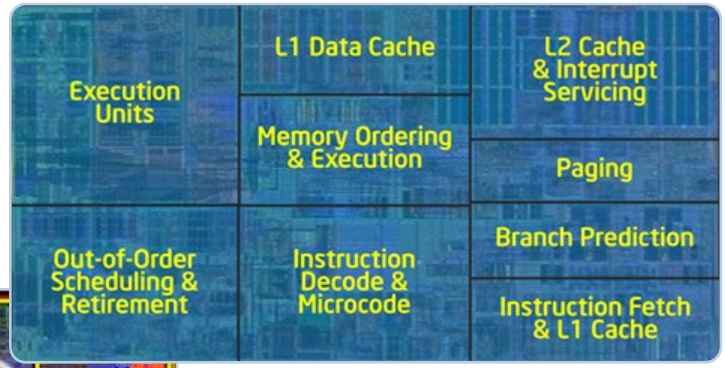
- *Les unités d'exécutions AGU (Address Generation Unit) sont chargées de calculer des adresses (modes d'adressage complexes) puis d'aller sauver ou chercher des données présentes en mémoire cache donnée. Ce sont les seules unités à posséder un chemin d'accès vers la mémoire donnée (jusqu'à 48o/cycle de bande passante).*

Les instructions et donc uops désirant réaliser un accès à la mémoire donnée amènent naturellement des latences. Les grandeurs données sont optimales et ne tiennent pas compte d'éventuels cache miss, défauts d'alignement, d'exceptions matérielles et de l'hyperthreading (partage des unités d'exécution) :

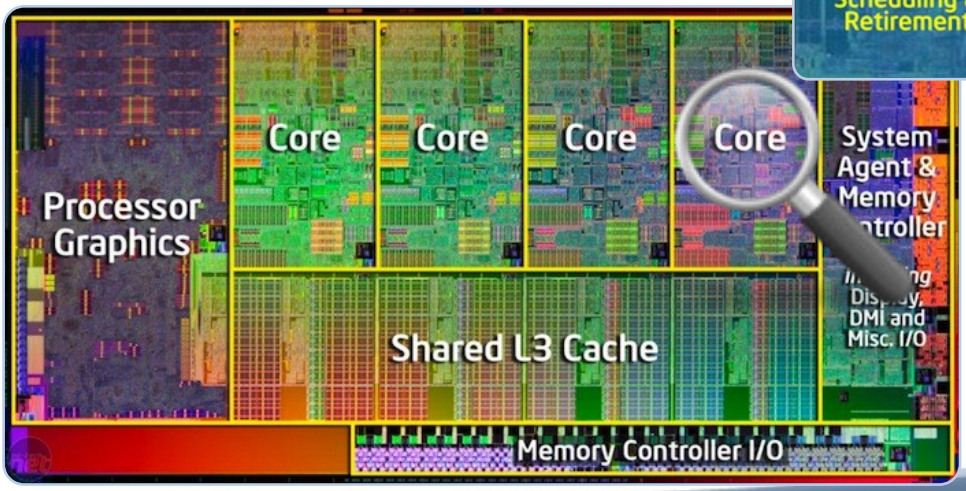
Instruction	Operands	uops fused	unfused uops					Latency	Reciprocal Throughput	Comments
			p0	p1	p5	p23	p4			
dpps	xmm, xmm, imm8	4	1	2	1	-	-	12	2	SSE4.1
	xmm, m128, imm8	6	5			1	-	12	4	

A titre indicatif, observons les empreintes silicium de chacun des étages ou entités précédemment présentées :

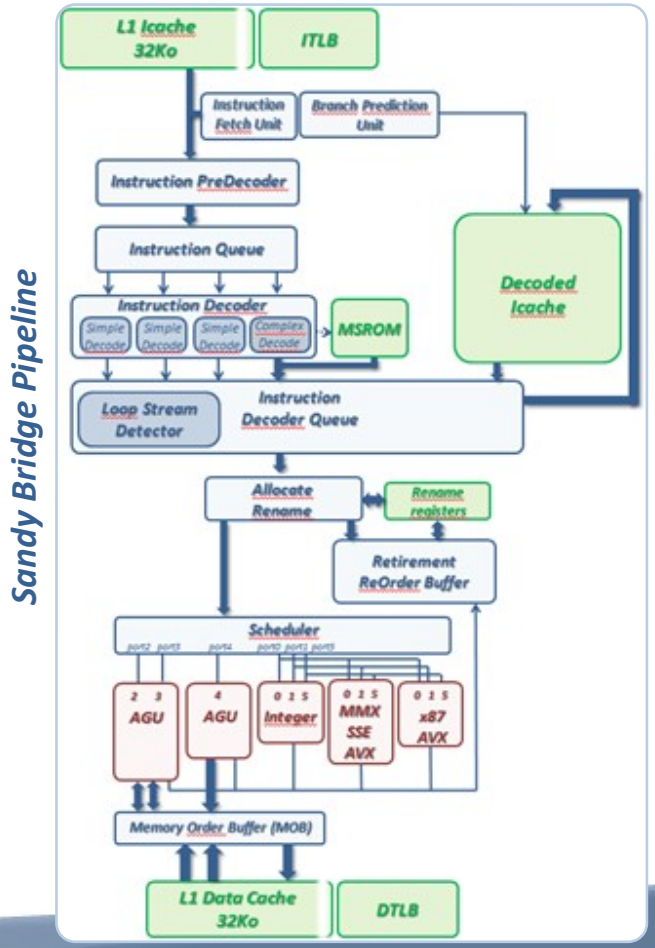
Sandy Bridge Core

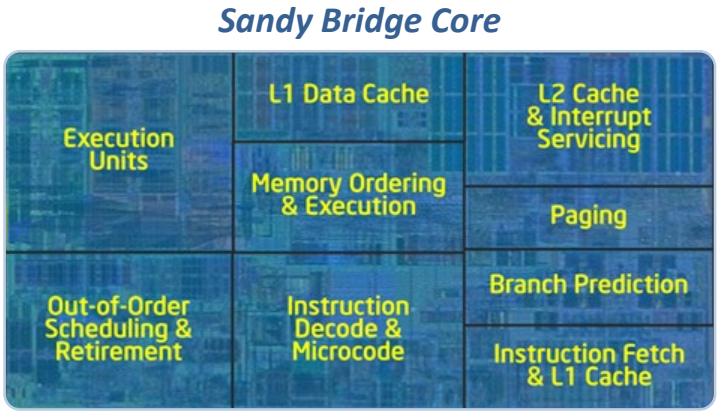


Sandy Bridge General Purpose Processor

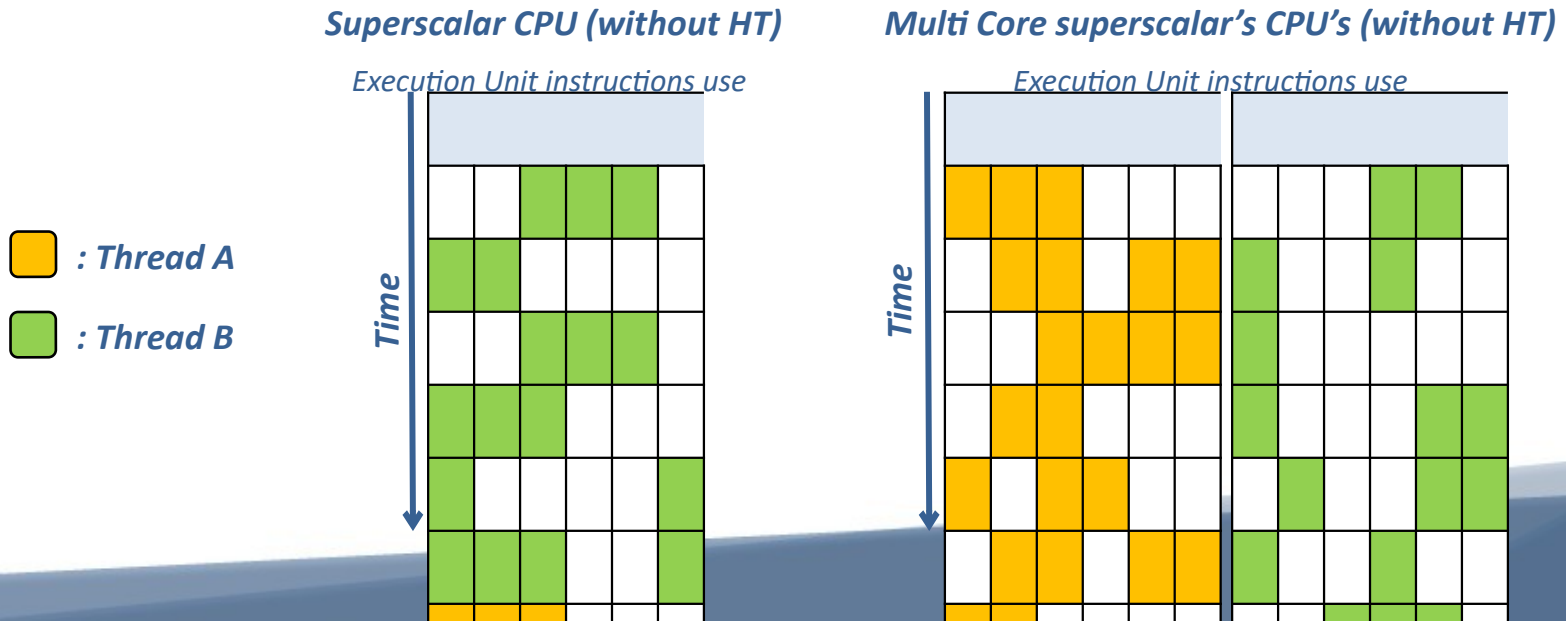


A titre indicatif, observons les empreintes silicium de chacun des étages ou entités précédemment présentées :

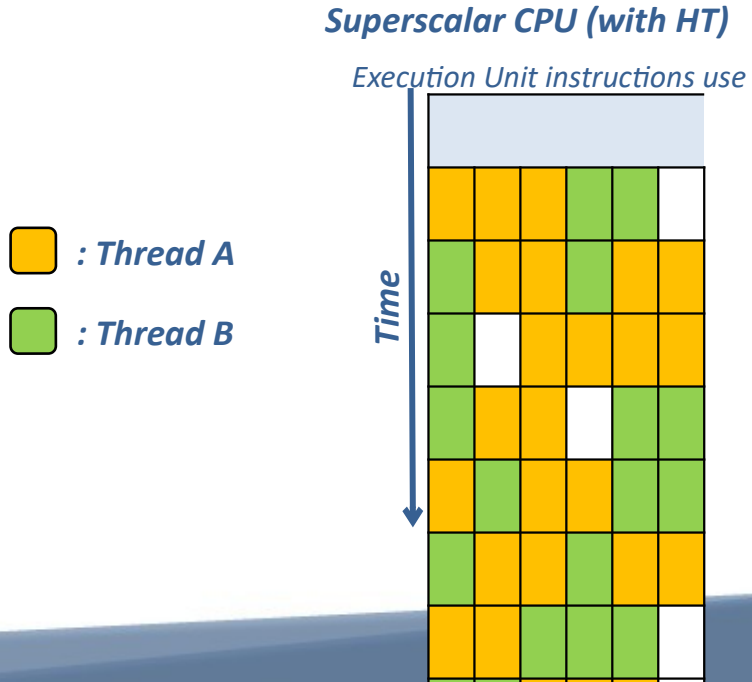


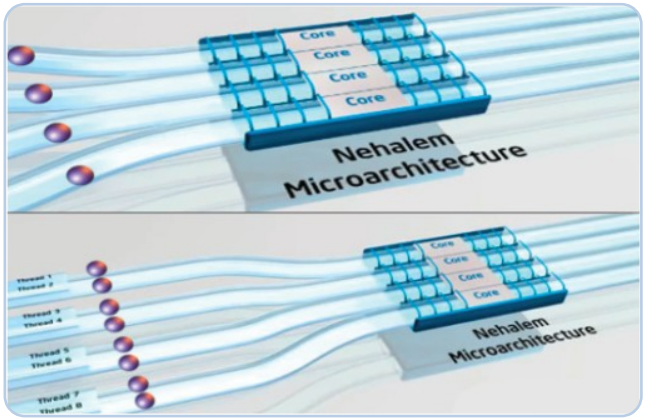


Une avancée technologique mal reconnue à l'époque de sa sortie en 2002 sur Pentium 4 mais néanmoins ré-implémenter depuis 2008 sur les architectures CoreiX et Atom est l'Hyper-Threading ou HT. Observons le principe de cette technologie n'exigeant que 5% de silicium supplémentaire mais pouvant dans certains cas améliorer les performances de 30% (applications multi-threads) :



Un processeur supportant l'hyper-threading peut exécuter jusqu'à 2 threads et est alors vu comme si il s'agissait de 2 CPU's distinct , nous parlons alors de CPU logique. Par exemple, pour un Corei7 famille Sandy Bridge 4 cores, le système d'exploitation voit 8 cœurs :



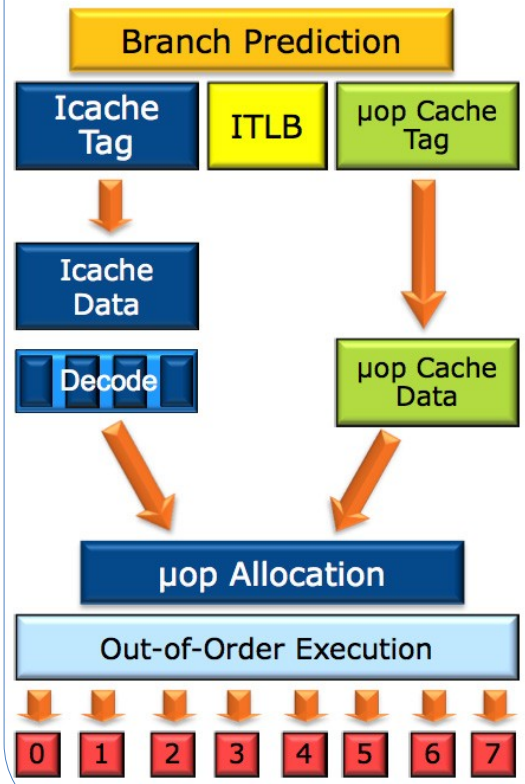


Observons les principales évolutions apportées avec la famille Haswell (4ieme génération Core) parue Juin 2013 :

- ***Cache :*** *L1D et L1I 64Ko 8-way associative, L2 1Mo 8-way associative, L3 jusqu'à 32Mo 16-way associative*
- ***Extensions jeu d'instructions :*** *AVX2 et FMA3 (Fused Multiple-Add) extension DSP au jeu d'instructions (Digital Signal Processing)*
- *Meilleure gestion d'énergie*
- ***Accélérateur Graphique :*** *support DirectX 11.1, OpenGL 4.0 et OpenCL 1.2*

Evolutions du pipeline matériel sur famille Haswell :

Haswell Core at a Glance



Next generation branch prediction
- Improves performance *and* saves wasted work

Improved front-end
- Initiate TLB and cache misses speculatively
- Handle cache misses in parallel to hide latency
- Leverages improved branch prediction

Deeper buffers
- Extract more instruction parallelism
- More resources when running a single thread

More execution units, shorter latencies
- Power down when not in use

More load/store bandwidth
- Better prefetching, better cache line split latency & throughput, double L2 bandwidth
- New modes save power without losing performance

No pipeline growth
- Same branch misprediction latency
- Same L1/L2 cache latency

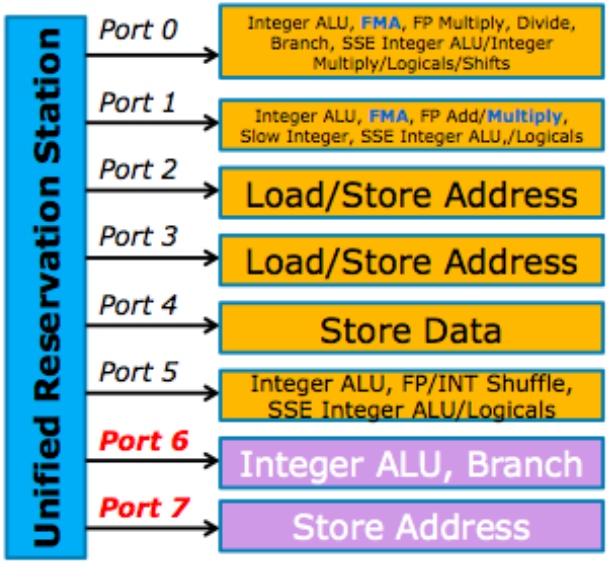
IDF2012 INTEL DEVELOPER FORUM

Intel® Microarchitecture (Haswell)

Evolutions du pipeline matériel sur famille Haswell :

Microarchitecture Enhancements

- **Energy-efficient performance**
  - Features to improve existing software
- **No change in key pipelines**
- **Example improvements**
  - Improved code fetch BW
  - Better branch prediction
  - Larger OOO window and corresponding structures
  - Increased throughput via 2 new dispatch ports
  - Larger L2 TLB
  - Lower virtualization latencies



Continue To Push Power-Efficient Performance Within the CPU

IDF2012
INTEL DEVELOPER FORUM

ENSICAEN
Computer Science

Merci de votre attention !