Departament d'Arquitectura de Computadors

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# Conceptes Avançats de Sistemes Operatius

Facultat d'Informàtica de Barcelona
Dept. d'Arquitectura de Computadors

Curs 2019/20 Q2

## Suport hardware

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

# Índex

# Índex

# Suport hardware

## Cap als spin locks

- Requereixen suport d'instruccions atòmiques
- Versió ideal, però **no funcional**:
  - L'execució de l'entrada a la regió crítica no és atòmica

```
int lock;
... Flux 1

while (lock==1) ;   //spin
lock = 1;
// regió crítica de codi
lock = 0;
```

```
... Flux 2

while (lock==1) ;   //spin
lock = 1;
// regió crítica de codi
lock = 0;
```

# Suport hardware

## Cap als spin locks

- Suport del compilador
  - Atribut `volatile`, indica que un altre flux pot estar accedint a la mateixa variable al mateix temps
  - Encara **no funcional**

```
volatile int lock;
... Flux 1

while (lock==1) ;   //spin
lock = 1;
// regió crítica de codi
lock = 0;
```

```
... Flux 2

while (lock==1) ;   //spin
lock = 1;
// regió crítica de codi
lock = 0;
```

# Suport hardware

## Suport per spin-locks

- A través d'intrínseques del compilador (gcc)

```
while (__sync_lock_test_and_set (&lock, 1)==1); //spin

   // regió crítica de codi

lock = 0;
```

# Suport hardware

## Suport per spin-locks

- Què genera el compilador?

```
bash-4.2$ objdump -d spin.o

0000000000000000 <f>:
   0:   55                          push   %rbp
   1:   48 89 e5                    mov    %rsp,%rbp
   4:   90                          nop
   5:   b8 01 00 00 00              mov    $0x1,%eax
   a:   87 05 00 00 00 00           xchg   %eax,0x0(%rip)  # 10 <f+0x10>
  10:   83 f8 01                    cmp    $0x1,%eax
  13:   74 f0                       je     5 <f+0x5>
  15:   bf 00 00 00 00              mov    $0x0,%edi   printf ("Hola\n");
  1a:   e8 00 00 00 00              callq  1f <f+0x1f>
  1f:   c7 05 00 00 00 00 00        movl   $0x0,0x0(%rip)  # 29 <f+0x29>
  26:   00 00 00
  29:   5d                          pop    %rbp
  2a:   c3                          retq
```

# Suport hardware

- Evitar la sobrecàrrega d'instruccions en els multicore d'Intel
  - Instrucció PAUSE
    - Fa que només hi hagi un "load" en vol
    - Facilita la invalidació (squash) de les instruccions quan el processador detecta una invalidació de la línia de cache

```
while (sync_var != value)
        asm __volatile__ (\pause");
```

- Evitar la sobrecàrrega del bus, per la transacció atòmica
  - Test, test-and-set

```
while (__sync_lock_test_and_set (&sync_var, BUSY)==BUSY)
        while (sync_var==BUSY) asm __volatile__ (\pause");
```

# Suport hardware

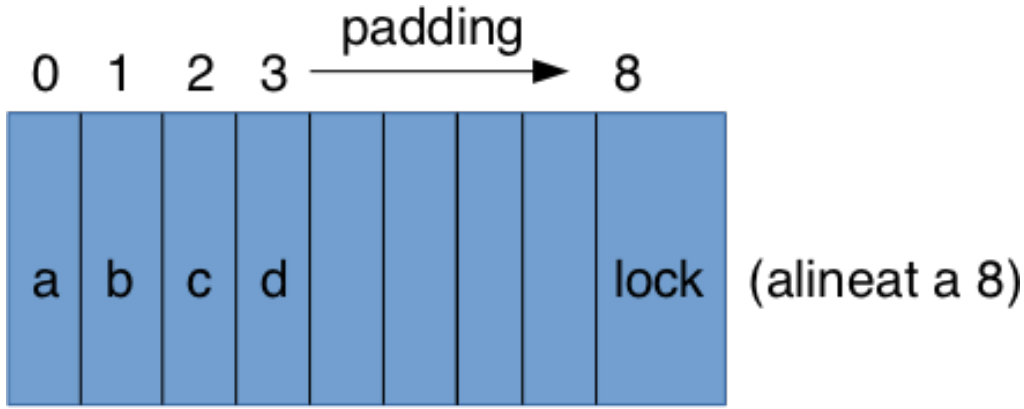
- Què genera el compilador?

```
0000000000000000 <f>:
   0:   ba 01 00 00 00          mov    $0x1,%edx
   5:   0f 1f 00                nopl   (%rax)
   8:   89 d0                   mov    %edx,%eax
   a:   87 05 00 00 00 00       xchg   %eax,0x0(%rip)        # 10 <f+0x10>
  10:   83 f8 01                cmp    $0x1,%eax
  13:   75 12                   jne    27 <f+0x27>
  15:   0f 1f 00                nopl   (%rax)
  18:   8b 05 00 00 00 00       mov    0x0(%rip),%eax        # 1e <f+0x1e>
  1e:   83 f8 01                cmp    $0x1,%eax
  21:   75 e5                   jne    8 <f+0x8>
  23:   f3 90                   pause
  25:   eb f1                   jmp    18 <f+0x18>
  27:   48 83 ec 08             sub    $0x8,%rsp
  2b:   bf 00 00 00 00          mov    $0x0,%edi
  30:   e8 00 00 00 00          callq  35 <f+0x35>
  35:   c7 05 00 00 00 00 00    movl   $0x0,0x0(%rip)        # 3f <f+0x3f>
  3c:   00 00 00
  3f:   48 83 c4 08             add    $0x8,%rsp
  43:   c3                      retq
```

# Suport hardware

- Deixar espai ("padding") entre variables de sincronització
  - Per evitar col·lisions a la cache
- Alinear variables almenys a la seva pròpia mida
  - Alinear implica "deixar padding, si cal"

```
volatile int lock __attribute__ ((aligned(128))); // alineat a 128
```



# Suport hardware

## Suport del compilador

- Funcions "intrinsic" per accedir a memòria
  - Inclouen una "barrera de memòria", memory barrier
    - El compilador no pot optimizar codi i moure les altres instruccions al voltant d'aquestes
    - Mantenen la semàntica del codi

```
TYPE __sync_fetch_and_add (TYPE *ptr, TYPE value, ...) + sub, or, and, xor, nand
TYPE __sync_add_and_fetch (TYPE *ptr, TYPE value, ...)  + ...
TYPE __sync_bool_compare_and_swap (TYPE *ptr, TYPE oldval, TYPE newval, ...)
TYPE __sync_val_compare_and_swap (TYPE *ptr, TYPE oldval, TYPE newval, ...)
TYPE __sync_lock_test_and_set (TYPE *ptr, TYPE value, ...)
void __sync_lock_release (TYPE *ptr, ...)
__sync_synchronize (...)
```

# Suport hardware

- `__sync_fetch_and_add (&val, 1)`
  `lock addl $0x1,2099117(%rip) # <val>`
- `__sync_lock_test_and_set (&lock, BUSY)`
  `xchg  %eax, 2099015(%rip) # <lock>`
- `__sync_synchronize ()`
  - No és necessària en Intel x86, x86_64
  - Necessària en altres arquitectures
    - IBM Power...

# Índex

# Suport hardware

## Comptadors hardware

- Permeten comptar events que passen durant l'execució
- Per codi d'usuari, sistema, excepció
- Limitacions:
  - Número petit de registres per comptar
    - Multiplexar events en els registres disponibles $\rightarrow$ SO
  - Registres "petits" (32bits)
    - Capturar excepcions en overflow $\rightarrow$ SO (signal)
  - Depenents de l'arquitectura i del processador
    - Processadors de la mateixa família tenen comptadors diferents

# Suport hardware

## Comptadors hardware

- Events interessants:
  - Cycles
  - Issued, graduated instructions
  - Issued, graduated loads/stores
  - Level N cache misses
  - TLB hits / misses
  - ALU/FPU progress cycles
  - Number of specific FP instructions issued, graduated
  - External intervention hits
  - External invalidations
  - and many more...
  - Deriving other metrics
    - Memory bandwidth
    - MFlops
    - IPC
    - ...

# Suport hardware

## Comptadors hardware

- Incorporar els comptadors en el context dels processos/fluxos
- Perfctr
  - Linux, incorporat en el sistema
  - Part de sistema, manteniment dels comptadors
  - Part d'usuari, eines per treure els comptadors i analitzar el seu significat
- PAPI: llibreria de suport per extreure la informació

# Suport hardware

## PAPI

- Eines
  - `papi_avail` mostra la disponibilitat dels comptadors estàndard
  - `papi_native_avail` comptadors nadius
  - `papi_mem_info` info. sobre TLB i caches
  - `papi_cost` cost de les crides de PAPI
  - `papi_event_chooser` indica la compatibilitat dels comptadors per anar alhora
  - ...

# Suport hardware

## Interfície PAPI

- `PAPI_start_counters (events, number)`
- `PAPI_read_counters (values, number)`
- `PAPI_accum_counters (values, number)`
- `PAPI_stop_counters (values, number)`
- `PAPI_flops (...)`
- `PAPI_create_eventset, PAPI_add_named_event, PAPI_overflow, PAPI_start, PAPI_stop`

# Suport hardware

- Informació del procés/flux
  - a través de crides a sistema
- Informació del processador
  - [Quin procés està executant actualment?]
  - Quan de temps està dedicant a executar codi
    - d'usuari?
    - de sistema?
    - en interrupció?

# Exemple: auto-avaluació

- Un programa pot treure els seus comptadors

```
void overflow_handler (int eventset, void * address, long long overflow_vector, void * context)
{
   printf ("%d: overflow occurred, overflow address %llx, distance %llx, vector %lld\n",
         sched_getcpu(), address, address - last_address, overflow_vector);
   last_address = address;
   num_overflows++;
}
res = PAPI_library_init(PAPI_VER_CURRENT);
if (res!=PAPI_VER_CURRENT) fprintf(stderr, "PAPI_library_init: (%d): %s\n", res, PAPI_strerror(res));
res = PAPI_create_eventset(&eventset);
res = PAPI_add_named_event(eventset, EVAL_COUNTER);
res = PAPI_event_name_to_code(EVAL_COUNTER, &code);
res = PAPI_overflow(eventset, code, EVAL_OVERFLOW, 0, overflow_handler);
res = PAPI_start(eventset);
/// codi a mesurar
res = PAPI_stop(eventset, NULL);
```

# Informació de l'entorn

## Fonts d'informació

- `/proc` informació general i debug de processos
- `/sys` informació especifica, components
- `/dev/pts` pseudoterminals en ús
- `/dev/input` dispositius d'entrada (ratolí, teclat...)
- `$ man mem`
  - `/dev/mem` memòria física de l'ordinador
  - `/dev/kmem` espai del kernel
  - `/dev/port` espai de ports d'E/S

# Per a la setmana vinent

- Provar els següents exemples:
  - Tema 1, Abstraccions: Demo de memòria compartida
  - Tema 2, Mach...: codi support hardware
    codi de threads i papi
  - Tema 3, Sincro...: codi de suport