

Processi, Thread, Multitasking, Multithreading

(Thread-Fu)



Quest'opera è distribuita con Licenza
Creative Commons Attribuzione - Non commerciale 4.0 Internazionale.

Prof. Giovambattista Ianni
Corso di Sistemi Operativi
Aggiornato all'A.A. 2022-23

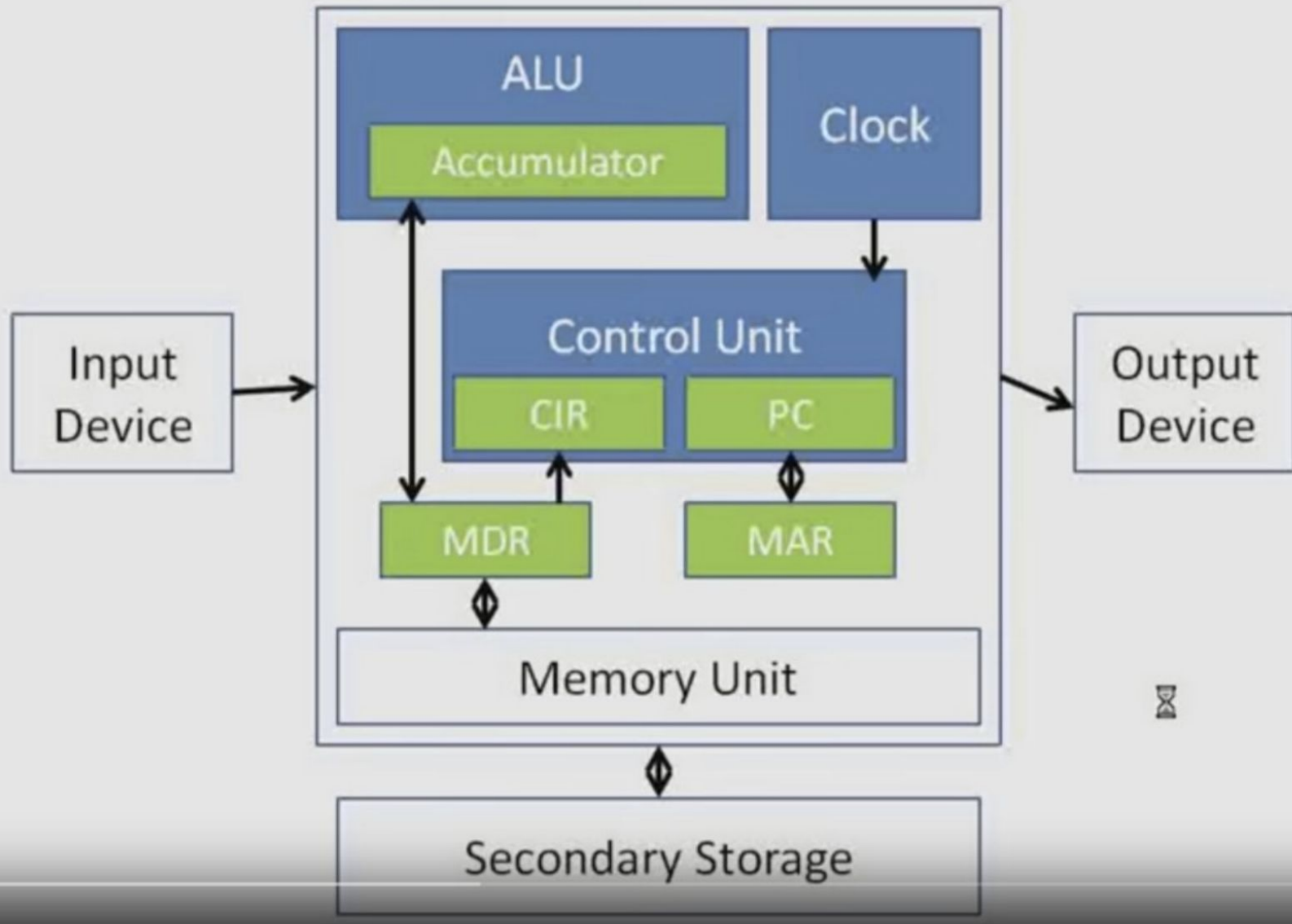
Ingredienti

- 1 CPU
- 1 Memoria RAM
- Tanti programmi che condividono la stessa memoria
- Esigenza di far girare più software
«*contemporaneamente*» (in **concorrenza**)

Processi e Thread

- Processo:
 - Associato a un singolo file binario eseguibile
 - Ha un suo stack e un suo spazio di memoria
 - Associato a un insieme di thread, che condividono la stessa memoria.
 - NON vede la memoria degli altri processi
- Thread:
 - ce ne sono uno o più di uno per processo (c'è almeno un *main thread*)
 - condividono le stesse risorse e *la stessa memoria*
 - hanno anche essi un proprio stack distinto

CPU



Anche la CPU ha della memoria, i **registri** (e la cache)

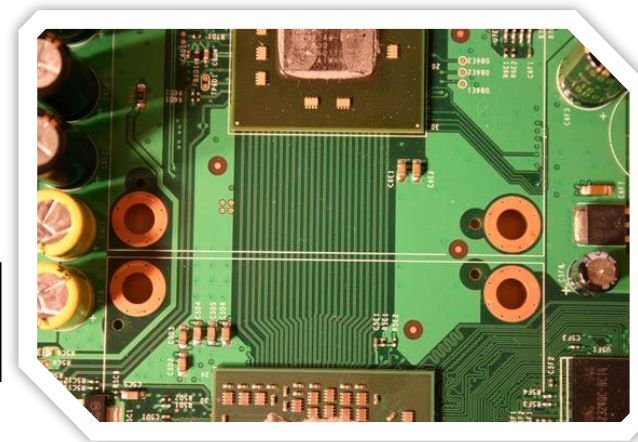
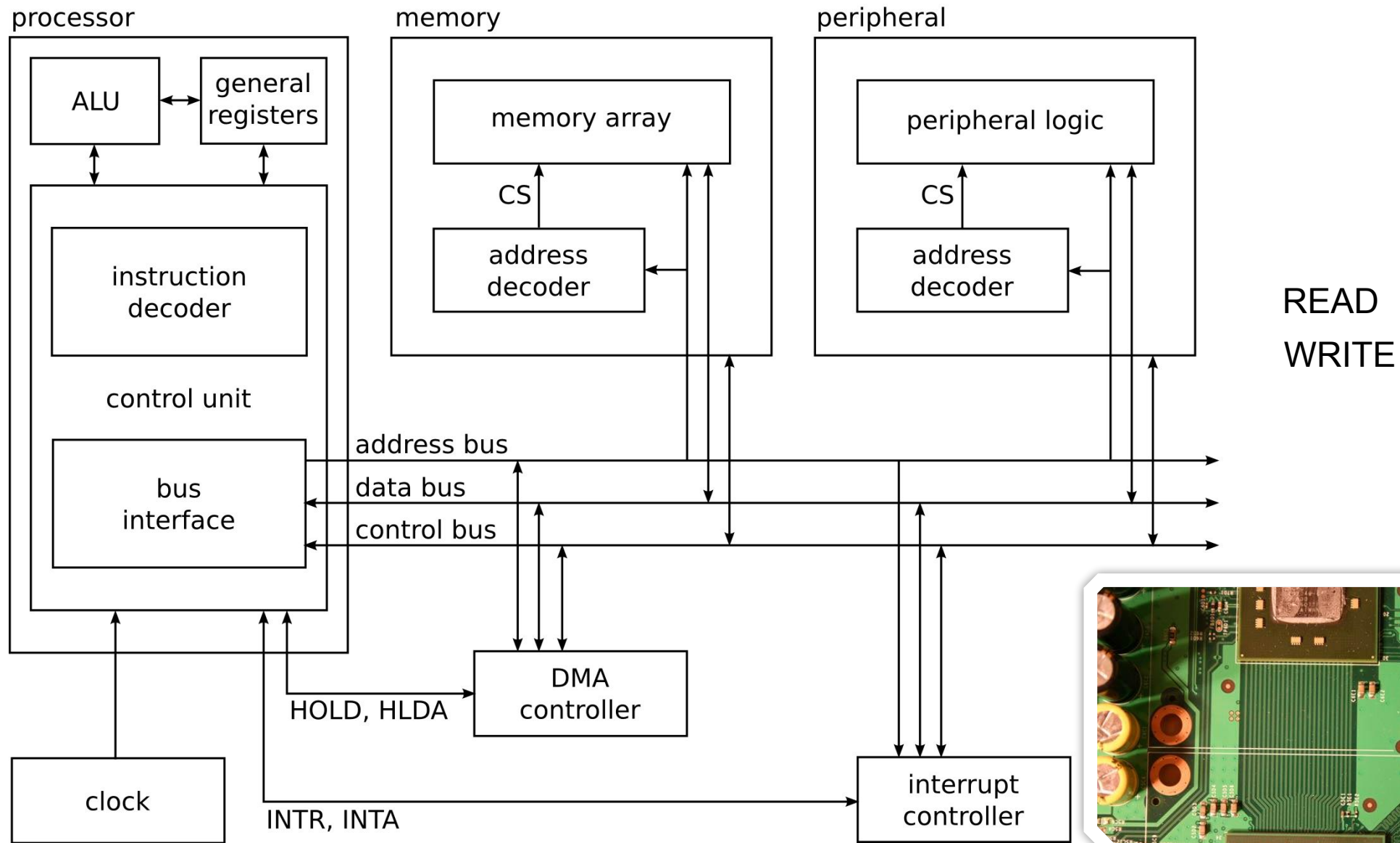
Per essere processati, i dati devono prima essere copiati nei registri

Il **program counter (PC)** contiene l'indirizzo della prossima istruzione da eseguire

Ciclo della CPU

- 1) Preleva istruzione dalla memoria (**fetch**)
- 2) Incrementa il PC
- 3) Esegue l'istruzione
- 4) Ripeti il processo

Alcune istruzioni possono modificare il PC



Ciclo di fetch esteso

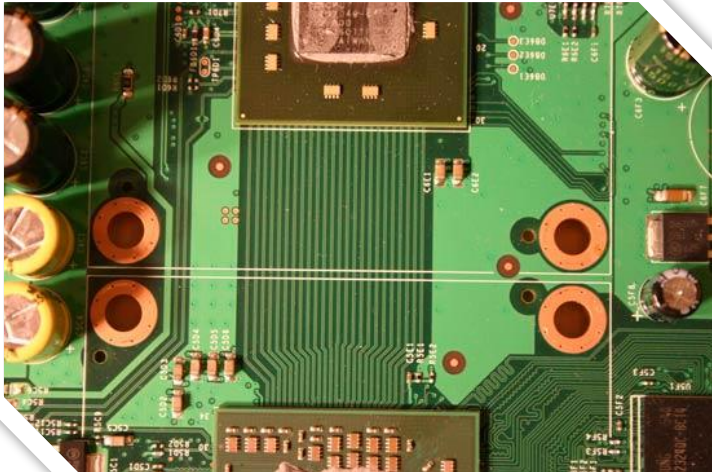
Classic fetch

0: IR \leftarrow Mem[PC]

PC \leftarrow PC+1

Esegui IR

Goto 0



Fetch con interrupt

0: IR \leftarrow Mem[PC]

PC \leftarrow PC+1

Esegui IR

If INTR == 1

oldPC \leftarrow PC

PC \leftarrow IRQR

Goto 0

IRQR = Interrupt request register: contiene l'indirizzo di memoria dove si trova la interrupt handling routine

Interrupt

- Usato dalle periferiche per notificare che un *evento urgente* deve essere gestito
- Esempi:
 - Clic del mouse □ I/O controller genera interrupt □ CPU gestisce interrupt □ evento clic finisce in coda eventi applicativi
 - Lettura da disco (vedi esempio su DMA)
 - Pressione di tasto
 - Terminazione di operazioni I/O in genere

DMA: Direct Memory Access

DMA: RAM accessibile da più entità, non solo la CPU

Esempio di accesso in scrittura in caso di RAM condivisa:

Ciclo di scrittura senza arbitraggio DMA

MAR \leftarrow addr

M[MAR] \leftarrow MBR

Ciclo di scrittura con arbitraggio DMA

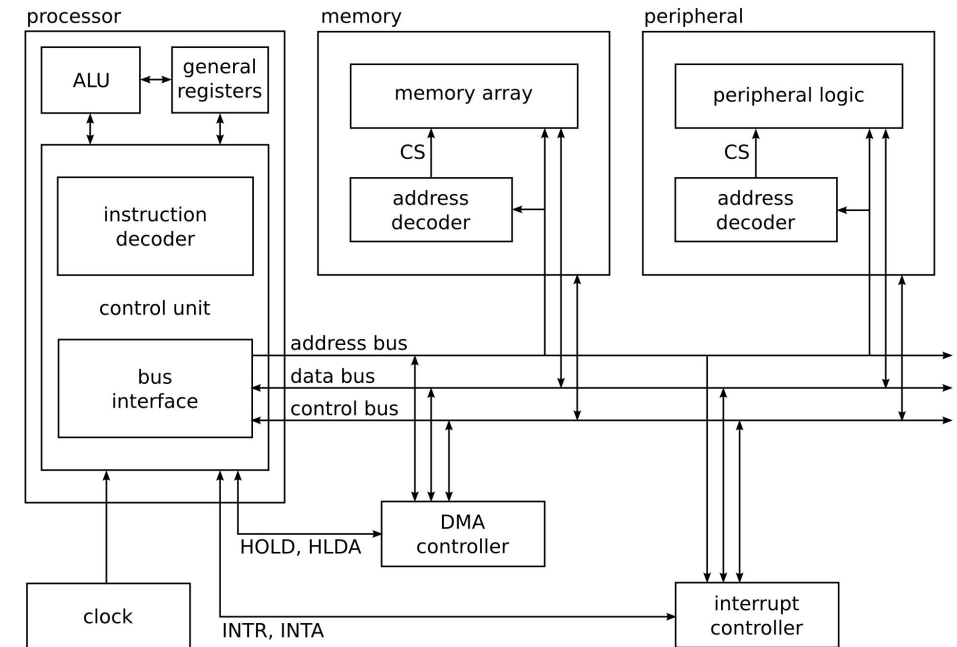
MAR \leftarrow addr

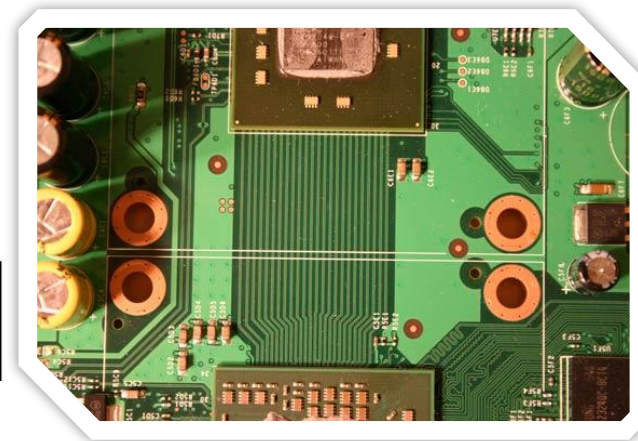
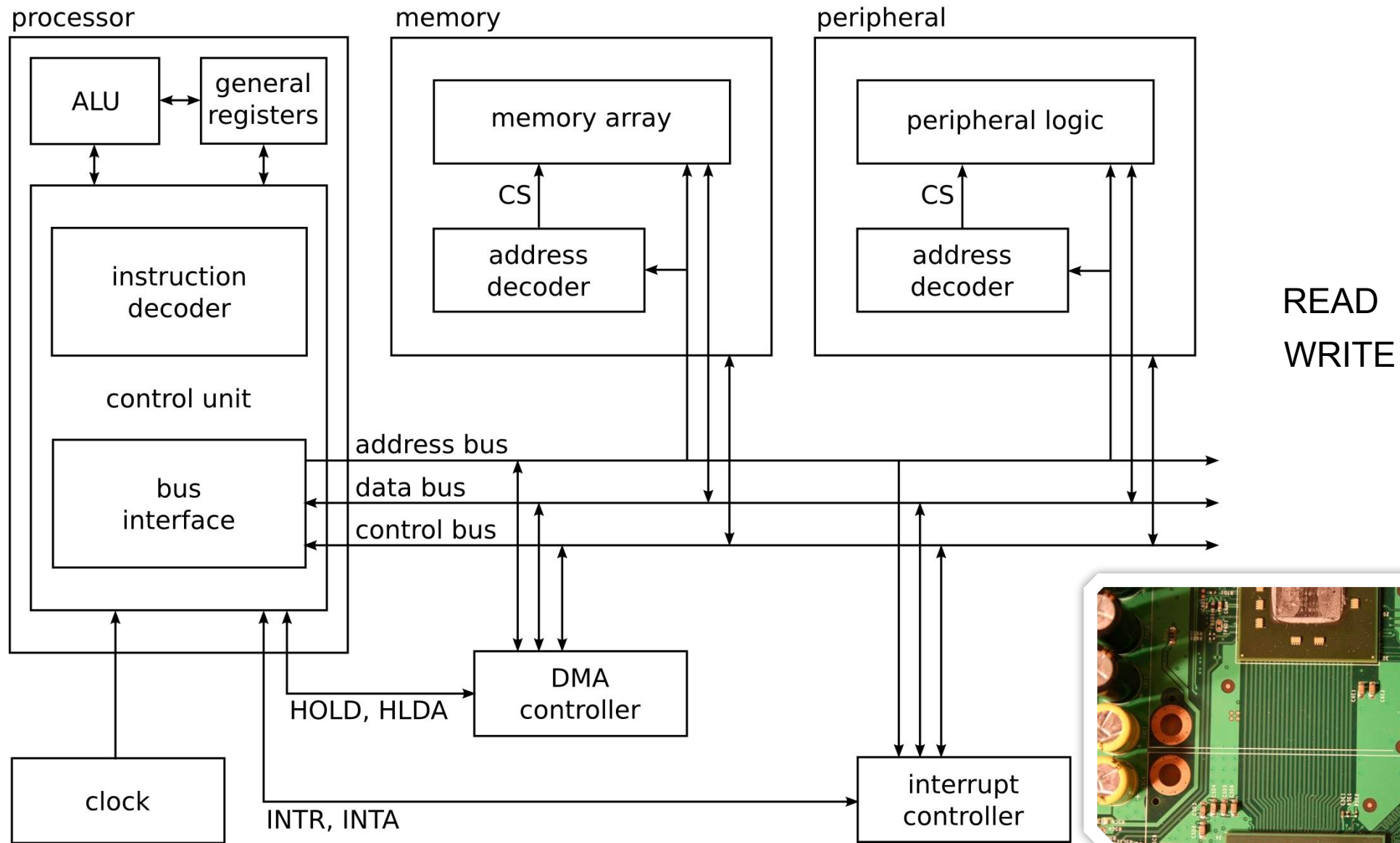
HOLD \leftarrow 1

0: **if** HLDA == 0

Goto 0

M[MAR] \leftarrow MBR





DMA transfer

- Esempio: disk DMA access
- A = indirizzo RAM dell'operazione DMA
- L = numero di byte da trasferire
- ID = ID della periferica coinvolta
- S = LBA address da leggere sul disco
- Operazioni di lettura
 - CPU imposta A, L, ID, S e invia un comando DMA Read
 - La periferica ID esegue la lettura e scrive su RAM in autonomia, a partire da indirizzo A
 - Al termine dell'operazione ID genera un interrupt
 - CPU gestisce questo interrupt

Multitasking Collaborativo - I

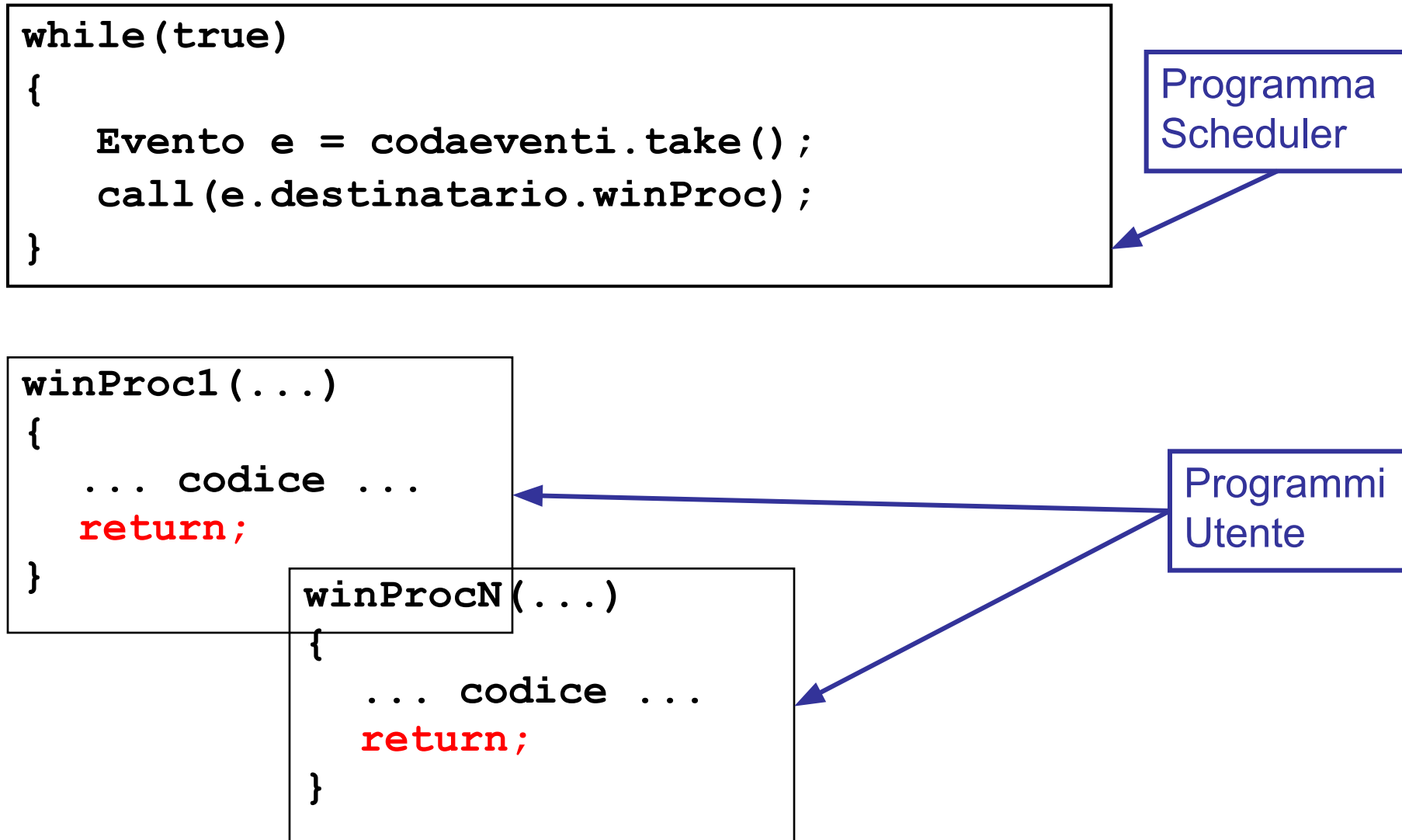
```
while(true)
{
    Evento e = codaeventi.take();
    call(e.destinatario.winProc);
}
```

Programma
Scheduler

```
winProc1(...)
{
    ... codice ...
    return;
}
```

```
winProcN(...)
{
    ... codice ...
    return;
}
```

Programmi
Utente



Multitasking collaborativo - II

- evento
 - qualcosa che è successa e che va gestita: tipicamente segnalato con un interrupt e poi depositato in una coda FIF
- codaeventi
 - Buffer FIFO di eventi che una certa applicazione deve gestire
- winprocA
 - Funzione associata a una certa applicazione A. Viene chiamata ogni volta che c'è da elaborare un evento destinato ad A. Il programmatore di A, definisce come gestire gli eventi.

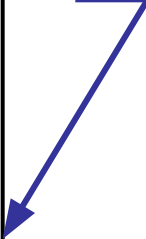
Multitasking collaborativo - III

- Pro:
 - Facile da capire/programmare
 - Comodo per multitasking tra task *I/O bound*
- Contro:
 - Una sola winproc che occupa tempo o va in loop infinito può bloccare tutto il SO/Applicazione/Tab del browser
 - Non sfrutta i core multipli
- Oggi, evoluto in *programmazione asincrona*, viene usato in:
 - Gestione eventi in Java
 - Gestione eventi in Javascript
 - Python Async I/O

Multitasking Non Collaborativo

```
while(true)
{
    Thread t = codaThreadPronti.take();
    impostaTimer();
    t.load();
    t.exec();
    t.save();
    inserisci t in
        codaThreadPronti
    oppure in listaThreadinWait;
}
```

Programma
Scheduler



Multitasking Non Collaborativo

- CodaThreadPronti
 - FIFO dei thread in stato di «Ready»
- ListaThreadInWait
 - Insieme dei thread in stato di «Wait»

load() e save()

- `save()` : scatta una “fotografia” del thread nel momento in cui si è sospeso, e la salva in memoria (TSS in Intel x86)
- `load()` : carica da memoria il TSS di un thread

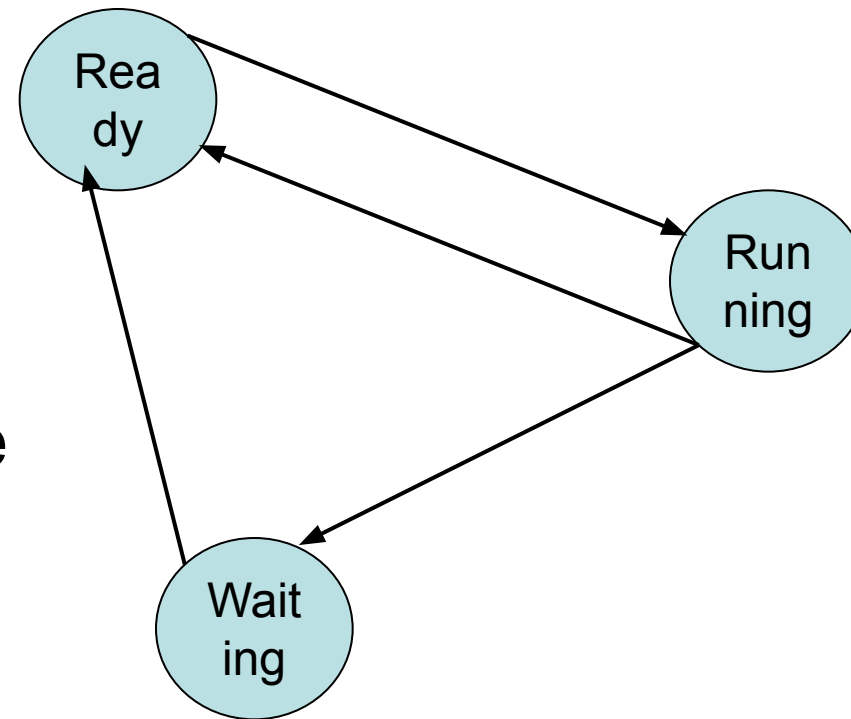
ARM	Description	x86
R0	General Purpose	EAX
R1-R5	General Purpose	EBX, ECX, EDX, ESI, EDI
R6-R10	General Purpose	–
R11 (FP)	Frame Pointer	EBP
R12	Intra Procedural Call	–
R13 (SP)	Stack Pointer	ESP
R14 (LR)	Link Register	–
R15 (PC)	<- Program Counter / Instruction Pointer ->	EIP
CPSR	Current Program State Register/Flags	EFLAGS

Che cosa avviene in exec()

- EIP registro analogo a PC
- assegna $EIP = TSS.EIP$ (analogo a $B \leftarrow TSS.EIP$)
- Esecuzione del codice del thread selezionato
- exec() Termina quando:
 1. scade il timer, oppure
 2. il thread va spontaneamente in stato di wait

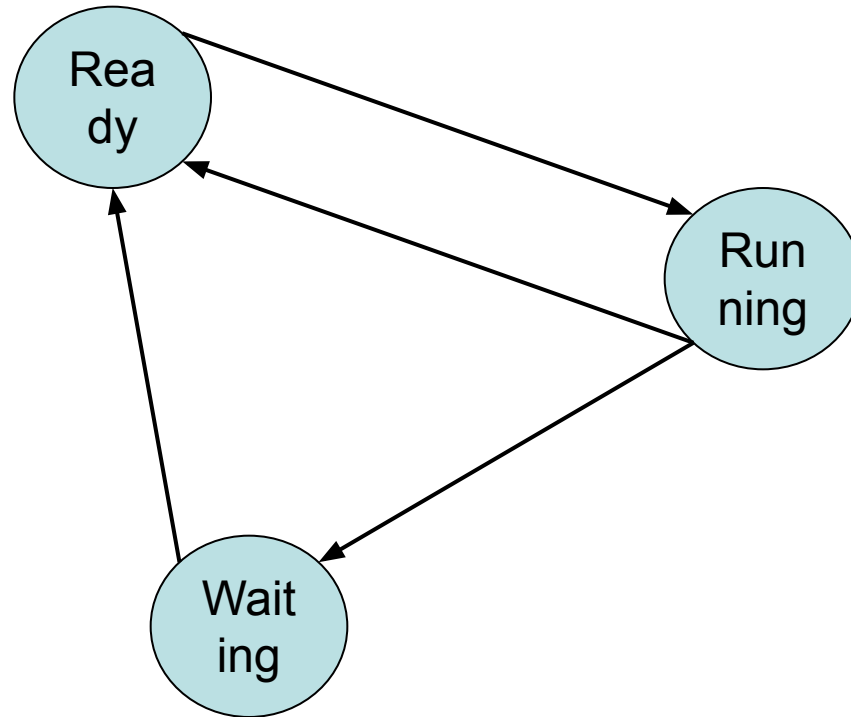
Stato di ogni thread

- Ready: pronto ad essere eseguito
- Running: in esecuzione
- Waiting: non può essere eseguito, in attesa di un evento esterno



Passaggi di stato

- Ready -> Running
 - Prelevato dalla coda ed eseguito
- Running -> Ready
 - Time out, cessazione volontaria
- Running -> Waiting
 - Invocazione di una call “bloccante”
- Waiting -> Ready
 - Ritorno da una call “bloccante”



Comportamento dei thread

- Thread I/O bound
 - Usa poco la CPU, spende molto tempo in attesa di eventi esterni. Si trova quasi sempre in stato di wait
- Thread CPU bound
 - Usa molto la CPU. Si trova quasi sempre in stato di wait
- Ovviamente un thread, nell'arco della propria esistenza, può cambiare comportamento a seconda del suo codice

Windows 2000-XP-Vista-7-8-10-11

- 32 Code
 - assegnazione del tempo a punti: 6 punti a testa = 1 timeslice
 - si guardano le code dalla 31 alla 0
 - **possibilità di *starvation***
 - meccanismi di promozione da una coda a un'altra
 - meccanismo di aumento dei punti
 - Il time-slicing è per thread non per processo!

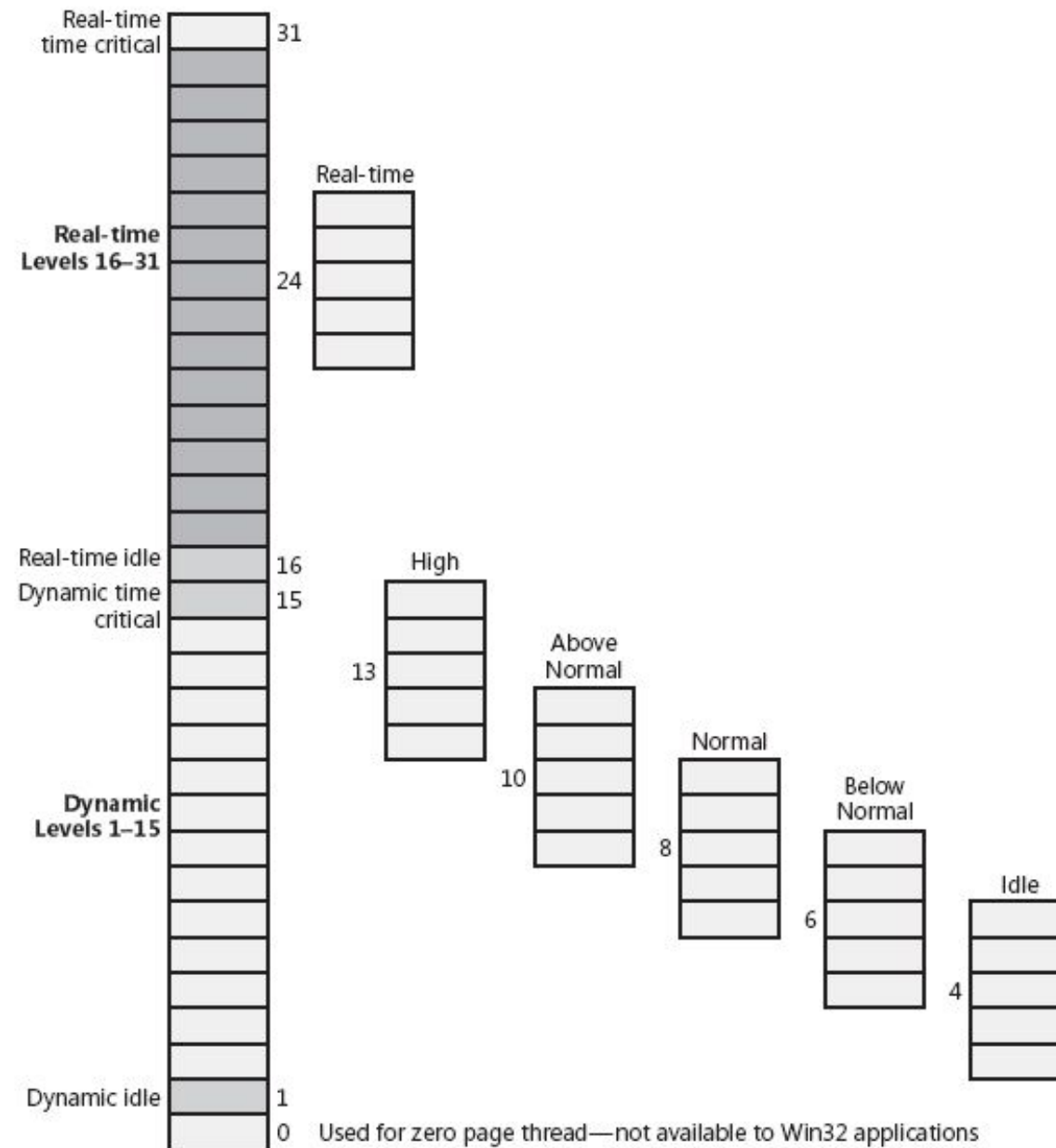


FIGURE 5-13 Mapping of Windows kernel priorities to the Windows API

Linux

- Normal process: priorità 100..139
 - nice level : -20 .. +19
- Real-time process: 1 .. 99
- Static and dynamic priority
- Scheduling types: SCHED_FIFO, SCHED_RR, **SCHED_OTHER** (CFS, the completely fair scheduler)

“I thread non servono a niente”

Scherzavo

Problemi di inconsistenza - 1

```
int posto[100];

int allocaposto(int p, int codiceutente)

{
    if (!posto[p])
        return posto[p] = codiceutente;
    else
        return 0;
}
```

Problemi di inconsistenza - 2

Assembly
x86

Posto 3

allocaposto(3,27049)

0

allocaposto(3,11051)

```
10: {  
    ...  
11: if (!posto[p])  
    mov     eax,dword ptr [ebp+8]  
    cmp     dword ptr [eax*4+4237A4h],0  
    jne     allocaposto+37h 0040b7e7)  
    return posto[p] = codiceutente;  
    mov     ecx,dword ptr [ebp+8]  
    mov     edx,dword ptr [ebp+0Ch]  
    mov     dword ptr [ecx*4+4237A4h],edx  
    mov     eax,dword ptr [ebp+0Ch]  
    jmp     allocaposto+39h 0040b7e9)  
13:     else  
14:         return 0;  
0040B7E7     xor     eax,eax  
15: }  
0040B7E9     pop     edi  
0040B7EA     pop     esi  
0040B7EB     pop     ebx  
0040B7EC     mov     esp,ebp  
0040B7EE     pop     ebp  
0040B7EF     ret
```

```
10: {  
    ...  
11: if (!posto[p])  
    mov     eax,dword ptr [ebp+8]  
    cmp     dword ptr [eax*4+4237A4h],0  
    jne     allocaposto+37h 0040b7e7)  
    return posto[p] = codiceutente;  
    mov     ecx,dword ptr [ebp+8]  
    mov     edx,dword ptr [ebp+0Ch]  
    mov     dword ptr [ecx*4+4237A4h],edx  
    mov     eax,dword ptr [ebp+0Ch]  
    jmp     allocaposto+39h 0040b7e9)  
13:     else  
14:         return 0;  
0040B7E7     xor     eax,eax  
15: }  
0040B7E9     pop     edi  
0040B7EA     pop     esi  
0040B7EB     pop     ebx  
0040B7EC     mov     esp,ebp  
0040B7EE     pop     ebp  
0040B7EF     ret
```

Problemi di inconsistenza - 2

Assembly
x86

Posto 3

allocaposto(3,27049)

0

allocaposto(3,11051)

```
10: {  
    ...  
11: if (!posto[p])  
    mov     eax,dword ptr [ebp+8]  
    cmp     dword ptr [eax*4+4237A4h],0  
    jne     allocaposto+37h 0040b7e7)  
    return posto[p] = codiceutente;  
    mov     ecx,dword ptr [ebp+8]  
    mov     edx,dword ptr [ebp+0Ch]  
    mov     dword ptr [ecx*4+4237A4h],edx  
    mov     eax,dword ptr [ebp+0Ch]  
    jmp     allocaposto+39h 0040b7e9)  
13:     else  
14:         return 0;  
0040B7E7    xor     eax,eax  
15: }  
0040B7E9    pop     edi  
0040B7EA    pop     esi  
0040B7EB    pop     ebx  
0040B7EC    mov     esp,ebp  
0040B7EE    pop     ebp  
0040B7EF    ret
```

```
10: {  
    ...  
11: if (!posto[p])  
    mov     eax,dword ptr [ebp+8]  
    cmp     dword ptr [eax*4+4237A4h],0  
    jne     allocaposto+37h 0040b7e7)  
    return posto[p] = codiceutente;  
    mov     ecx,dword ptr [ebp+8]  
    mov     edx,dword ptr [ebp+0Ch]  
    mov     dword ptr [ecx*4+4237A4h],edx  
    mov     eax,dword ptr [ebp+0Ch]  
    jmp     allocaposto+39h 0040b7e9)  
13:     else  
14:         return 0;  
0040B7E7    xor     eax,eax  
15: }  
0040B7E9    pop     edi  
0040B7EA    pop     esi  
0040B7EB    pop     ebx  
0040B7EC    mov     esp,ebp  
0040B7EE    pop     ebp  
0040B7EF    ret
```

Problemi di inconsistenza - 2

Assembly
x86

Posto 3

allocaposto(3,27049)

27049

allocaposto(3,11051)

```
10: {  
    ...  
11: if (!posto[p])  
    mov     eax,dword ptr [ebp+8]  
    cmp     dword ptr [eax*4+4237A4h],0  
    jne     allocaposto+37h 0040b7e7)  
    return posto[p] = codiceutente;  
    mov     ecx,dword ptr [ebp+8]  
    mov     edx,dword ptr [ebp+0Ch]  
    mov     dword ptr [ecx*4+4237A4h],edx  
    mov     eax,dword ptr [ebp+0Ch]  
    jmp     allocaposto+39h 0040b7e9)  
13:     else  
14:         return 0;  
0040B7E7    xor     eax,eax  
15: }  
0040B7E9    pop     edi  
0040B7EA    pop     esi  
0040B7EB    pop     ebx  
0040B7EC    mov     esp,ebp  
0040B7EE    pop     ebp  
0040B7EF    ret
```

```
10: {  
    ...  
11: if (!posto[p])  
    mov     eax,dword ptr [ebp+8]  
    cmp     dword ptr [eax*4+4237A4h],0  
    jne     allocaposto+37h 0040b7e7)  
    return posto[p] = codiceutente;  
    mov     ecx,dword ptr [ebp+8]  
    mov     edx,dword ptr [ebp+0Ch]  
    mov     dword ptr [ecx*4+4237A4h],edx  
    mov     eax,dword ptr [ebp+0Ch]  
    jmp     allocaposto+39h 0040b7e9)  
13:     else  
14:         return 0;  
0040B7E7    xor     eax,eax  
15: }  
0040B7E9    pop     edi  
0040B7EA    pop     esi  
0040B7EB    pop     ebx  
0040B7EC    mov     esp,ebp  
0040B7EE    pop     ebp  
0040B7EF    ret
```

Problemi di inconsistenza - 2

Assembly
x86

Posto 3

allocaposto(3,27049)

27049

allocaposto(3,11051)

```
10: {  
    ...  
11: if (!posto[p])  
    mov     eax,dword ptr [ebp+8]  
    cmp     dword ptr [eax*4+4237A4h],0  
    jne     allocaposto+37h 0040b7e7)  
    return posto[p] = codiceutente;  
    mov     ecx,dword ptr [ebp+8]  
    mov     edx,dword ptr [ebp+0Ch]  
    mov     dword ptr [ecx*4+4237A4h],edx  
    mov     eax,dword ptr [ebp+0Ch]  
    jmp     allocaposto+39h 0040b7e9)  
13:     else  
14:         return 0;  
0040B7E7 xor     eax,eax  
15: }  
0040B7E9 pop     edi  
0040B7EA pop     esi  
0040B7EB pop     ebx  
0040B7EC mov     esp,ebp  
0040B7EE pop     ebp  
0040B7EF ret
```

```
10: {  
    ...  
11: if (!posto[p])  
    mov     eax,dword ptr [ebp+8]  
    cmp     dword ptr [eax*4+4237A4h],0  
    jne     allocaposto+37h 0040b7e7)  
    return posto[p] = codiceutente;  
    mov     ecx,dword ptr [ebp+8]  
    mov     edx,dword ptr [ebp+0Ch]  
    mov     dword ptr [ecx*4+4237A4h],edx  
    mov     eax,dword ptr [ebp+0Ch]  
    jmp     allocaposto+39h 0040b7e9)  
13:     else  
14:         return 0;  
0040B7E7 xor     eax,eax  
15: }  
0040B7E9 pop     edi  
0040B7EA pop     esi  
0040B7EB pop     ebx  
0040B7EC mov     esp,ebp  
0040B7EE pop     ebp  
0040B7EF ret
```

Problemi di inconsistenza - 2

Assembly
x86

Posto 3

allocaposto(3,27049)

11051

allocaposto(3,11051)

```
10: {  
    ...  
11: if (!posto[p])  
    mov     eax,dword ptr [ebp+8]  
    cmp     dword ptr [eax*4+4237A4h],0  
    jne     allocaposto+37h 0040b7e7)  
    return posto[p] = codiceutente;  
    mov     ecx,dword ptr [ebp+8]  
    mov     edx,dword ptr [ebp+0Ch]  
    mov     dword ptr [ecx*4+4237A4h],edx  
    mov     eax,dword ptr [ebp+0Ch]  
    jmp     allocaposto+39h 0040b7e9)  
13:     else  
14:         return 0;  
0040B7E7    xor     eax,eax  
15: }  
0040B7E9    pop     edi  
0040B7EA    pop     esi  
0040B7EB    pop     ebx  
0040B7EC    mov     esp,ebp  
0040B7EE    pop     ebp  
0040B7EF    ret
```

```
10: {  
    ...  
11: if (!posto[p])  
    mov     eax,dword ptr [ebp+8]  
    cmp     dword ptr [eax*4+4237A4h],0  
    jne     allocaposto+37h 0040b7e7)  
    return posto[p] = codiceutente;  
    mov     ecx,dword ptr [ebp+8]  
    mov     edx,dword ptr [ebp+0Ch]  
    mov     dword ptr [ecx*4+4237A4h],edx  
    mov     eax,dword ptr [ebp+0Ch]  
    jmp     allocaposto+39h 0040b7e9)  
13:     else  
14:         return 0;  
0040B7E7    xor     eax,eax  
15: }  
0040B7E9    pop     edi  
0040B7EA    pop     esi  
0040B7EB    pop     ebx  
0040B7EC    mov     esp,ebp  
0040B7EE    pop     ebp  
0040B7EF    ret
```

```

    allocaposto(3,27049);
40070c:    528d3521    mov     w1, #0x69a9           // #27049
400710:    52800060    mov     w0, #0x3             // #3
400714:    97ffffe6    bl      4006ac <_Z11allocapostoi>
    allocaposto(3,11051);
400718:    52856561    mov     w1, #0x2b2b           // #11051
40071c:    52800060    mov     w0, #0x3             // #3
400720:    97ffffe3    bl      4006ac <_Z11allocapostoi>

```

Problemi di inconsistenza - 2

Assembly
ARM64

Posto 3

allocaposto(3,27049)

0

allocaposto(3,11051)

```
10: {  
    ...  
    if (!posto[p])  
        4006b8: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006bc: add     x0, x0, #0x9c0  
        4006c0: ldrsw   x1, [sp, #12]  
        4006c4: ldr     w0, [x0, x1, lsl #2]  
        4006c8: cmp     w0, #0x0  
        4006cc: b.ne    4006f8 <_Z11allocapostoi+0x4c>  
        return posto[p] = codiceutente;  
        4006d0: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006d4: add     x0, x0, #0x9c0  
        4006d8: ldrsw   x1, [sp, #12]  
        4006dc: ldr     w2, [sp, #8]  
        4006e0: str     w2, [x0, x1, lsl #2]  
        4006e4: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006e8: add     x0, x0, #0x9c0  
        4006ec: ldrsw   x1, [sp, #12]  
        4006f0: ldr     w0, [x0, x1, lsl #2]  
        4006f4: b       4006fc <_Z11allocapostoi+0x50>  
    else  
        return 0;  
        4006f8: mov     w0, #0x0  
}  
4006fc: add     sp, sp, #0x10  
400700: ret
```

```
10: {  
    ...  
    if (!posto[p])  
        4006b8: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006bc: add     x0, x0, #0x9c0  
        4006c0: ldrsw   x1, [sp, #12]  
        4006c4: ldr     w0, [x0, x1, lsl #2]  
        4006c8: cmp     w0, #0x0  
        4006cc: b.ne    4006f8 <_Z11allocapostoi+0x4c>  
        return posto[p] = codiceutente;  
        4006d0: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006d4: add     x0, x0, #0x9c0  
        4006d8: ldrsw   x1, [sp, #12]  
        4006dc: ldr     w2, [sp, #8]  
        4006e0: str     w2, [x0, x1, lsl #2]  
        4006e4: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006e8: add     x0, x0, #0x9c0  
        4006ec: ldrsw   x1, [sp, #12]  
        4006f0: ldr     w0, [x0, x1, lsl #2]  
        4006f4: b       4006fc <_Z11allocapostoi+0x50>  
    else  
        return 0;  
        4006f8: mov     w0, #0x0  
}  
4006fc: add     sp, sp, #0x10  
400700: ret
```


Problemi di inconsistenza - 2

Assembly
ARM64

Posto 3

allocaposto(3,27049)

0

allocaposto(3,11051)

```
10: {
    ...
    if (!posto[p])
        4006b8: adrp    x0, 489000 <_dl_main_map+0xc0>
        4006bc: add     x0, x0, #0x9c0
        4006c0: ldrsw   x1, [sp, #12]
        4006c4: ldr     w0, [x0, x1, lsl #2]
        4006c8: cmp     w0, #0x0
        4006cc: b.ne    4006f8 <_Z11allocapostoi+0x4c>
        return posto[p] = codiceutente;
        4006d0: adrp    x0, 489000 <_dl_main_map+0xc0>
        4006d4: add     x0, x0, #0x9c0
        4006d8: ldrsw   x1, [sp, #12]
        4006dc: ldr     w2, [sp, #8]
        4006e0: str     w2, [x0, x1, lsl #2]
        4006e4: adrp    x0, 489000 <_dl_main_map+0xc0>
        4006e8: add     x0, x0, #0x9c0
        4006ec: ldrsw   x1, [sp, #12]
        4006f0: ldr     w0, [x0, x1, lsl #2]
        4006f4: b       4006fc <_Z11allocapostoi+0x50>
    else
        return 0;
        4006f8: mov     w0, #0x0
}
4006fc: add     sp, sp, #0x10
400700: ret
```

```
10: {
    ...
    if (!posto[p])
        4006b8: adrp    x0, 489000 <_dl_main_map+0xc0>
        4006bc: add     x0, x0, #0x9c0
        4006c0: ldrsw   x1, [sp, #12]
        4006c4: ldr     w0, [x0, x1, lsl #2]
        4006c8: cmp     w0, #0x0
        4006cc: b.ne    4006f8 <_Z11allocapostoi+0x4c>
        return posto[p] = codiceutente;
        4006d0: adrp    x0, 489000 <_dl_main_map+0xc0>
        4006d4: add     x0, x0, #0x9c0
        4006d8: ldrsw   x1, [sp, #12]
        4006dc: ldr     w2, [sp, #8]
        4006e0: str     w2, [x0, x1, lsl #2]
        4006e4: adrp    x0, 489000 <_dl_main_map+0xc0>
        4006e8: add     x0, x0, #0x9c0
        4006ec: ldrsw   x1, [sp, #12]
        4006f0: ldr     w0, [x0, x1, lsl #2]
        4006f4: b       4006fc <_Z11allocapostoi+0x50>
    else
        return 0;
        4006f8: mov     w0, #0x0
}
4006fc: add     sp, sp, #0x10
400700: ret
```

Problemi di inconsistenza - 2

Assembly
ARM64

Posto 3

allocaposto(3,27049)

27049

allocaposto(3,11051)

```
10: {  
    ...  
    if (!posto[p])  
        4006b8: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006bc: add     x0, x0, #0x9c0  
        4006c0: ldrsw   x1, [sp, #12]  
        4006c4: ldr     w0, [x0, x1, lsl #2]  
        4006c8: cmp     w0, #0x0  
        4006cc: b.ne    4006f8 <_Z11allocapostoi+0x4c>  
        return posto[p] = codiceutente;  
        4006d0: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006d4: add     x0, x0, #0x9c0  
        4006d8: ldrsw   x1, [sp, #12]  
        4006dc: ldr     w2, [sp, #8]  
        4006e0: str     w2, [x0, x1, lsl #2]  
        4006e4: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006e8: add     x0, x0, #0x9c0  
        4006ec: ldrsw   x1, [sp, #12]  
        4006f0: ldr     w0, [x0, x1, lsl #2]  
        4006f4: b       4006fc <_Z11allocapostoi+0x50>  
    else  
        return 0;  
        4006f8: mov     w0, #0x0  
}  
4006fc: add     sp, sp, #0x10  
400700: ret
```

```
10: {  
    ...  
    if (!posto[p])  
        4006b8: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006bc: add     x0, x0, #0x9c0  
        4006c0: ldrsw   x1, [sp, #12]  
        4006c4: ldr     w0, [x0, x1, lsl #2]  
        4006c8: cmp     w0, #0x0  
        4006cc: b.ne    4006f8 <_Z11allocapostoi+0x4c>  
        return posto[p] = codiceutente;  
        4006d0: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006d4: add     x0, x0, #0x9c0  
        4006d8: ldrsw   x1, [sp, #12]  
        4006dc: ldr     w2, [sp, #8]  
        4006e0: str     w2, [x0, x1, lsl #2]  
        4006e4: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006e8: add     x0, x0, #0x9c0  
        4006ec: ldrsw   x1, [sp, #12]  
        4006f0: ldr     w0, [x0, x1, lsl #2]  
        4006f4: b       4006fc <_Z11allocapostoi+0x50>  
    else  
        return 0;  
        4006f8: mov     w0, #0x0  
}  
4006fc: add     sp, sp, #0x10  
400700: ret
```

Problemi di inconsistenza - 2

Assembly
ARM64

Posto 3

allocaposto(3,27049)

27049

allocaposto(3,11051)

```
10: {  
    ...  
    if (!posto[p])  
        4006b8: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006bc: add     x0, x0, #0x9c0  
        4006c0: ldrsw   x1, [sp, #12]  
        4006c4: ldr     w0, [x0, x1, lsl #2]  
        4006c8: cmp     w0, #0x0  
        4006cc: b.ne    4006f8 <_Z11allocapostoi+0x4c>  
        return posto[p] = codiceutente;  
        4006d0: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006d4: add     x0, x0, #0x9c0  
        4006d8: ldrsw   x1, [sp, #12]  
        4006dc: ldr     w2, [sp, #8]  
        4006e0: str     w2, [x0, x1, lsl #2]  
        4006e4: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006e8: add     x0, x0, #0x9c0  
        4006ec: ldrsw   x1, [sp, #12]  
        4006f0: ldr     w0, [x0, x1, lsl #2]  
        4006f4: b       4006fc <_Z11allocapostoi+0x50>  
    else  
        return 0;  
        4006f8: mov     w0, #0x0  
}  
4006fc: add     sp, sp, #0x10  
400700: ret
```

```
10: {  
    ...  
    if (!posto[p])  
        4006b8: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006bc: add     x0, x0, #0x9c0  
        4006c0: ldrsw   x1, [sp, #12]  
        4006c4: ldr     w0, [x0, x1, lsl #2]  
        4006c8: cmp     w0, #0x0  
        4006cc: b.ne    4006f8 <_Z11allocapostoi+0x4c>  
        return posto[p] = codiceutente;  
        4006d0: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006d4: add     x0, x0, #0x9c0  
        4006d8: ldrsw   x1, [sp, #12]  
        4006dc: ldr     w2, [sp, #8]  
        4006e0: str     w2, [x0, x1, lsl #2]  
        4006e4: adrp    x0, 489000 <_dl_main_map+0xc0>  
        4006e8: add     x0, x0, #0x9c0  
        4006ec: ldrsw   x1, [sp, #12]  
        4006f0: ldr     w0, [x0, x1, lsl #2]  
        4006f4: b       4006fc <_Z11allocapostoi+0x50>  
    else  
        return 0;  
        4006f8: mov     w0, #0x0  
}  
4006fc: add     sp, sp, #0x10  
400700: ret
```

Voglio vedere il codice misto anche io!

```
gcc -g miosorgente.cpp  
objdump -S miobinario
```

Per aarch64 usare l'accoppiata:

```
aarch64-linux-gnu-gcc  
aarch64-linux-gnu-objdump
```

Race condition

- Situazione in cui due o più thread “competono” senza controllo o disciplina nel modificare o leggere contemporaneamente gli stessi dati
 - RISULTATI IMPREDICIBILI
- Un software che consente race condition incontrollate non è “Thread-safe”
- Un programmatore che non sa gestire le race condition non è un bravo programmatore

Esempi di race condition

```
def bonifico(A : conto, B : conto, s : int):  
  
    A.saldo += s  
    B.saldo -= s
```

RARO? Seriamente, RARO?

PALCOSCENICO

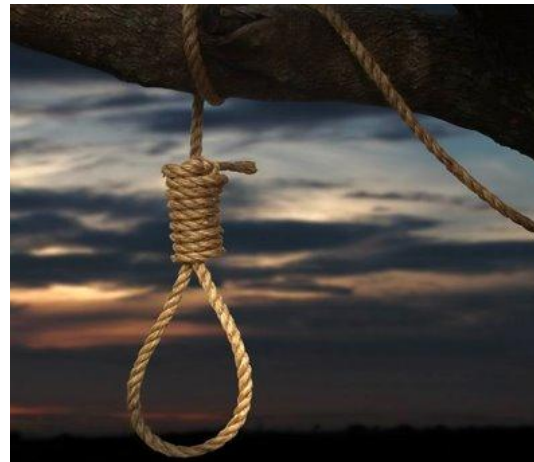
Posti Riservati per la Vendita on-Line. Posti Diversi potrebbero essere disponibili nei punti Vendita.

La prenotazione verrà effettuata alla fine del pagamento, e confermata tramite e-mail.

Ricordiamo quindi che posti liberi in fase di ordine potrebbero non esserlo al momento dell'effettiva registrazione. In questo (peraltro rarissimo caso), si dovrà chiedere il rimborso, e ripetere la procedura scegliendo nuovi posti.

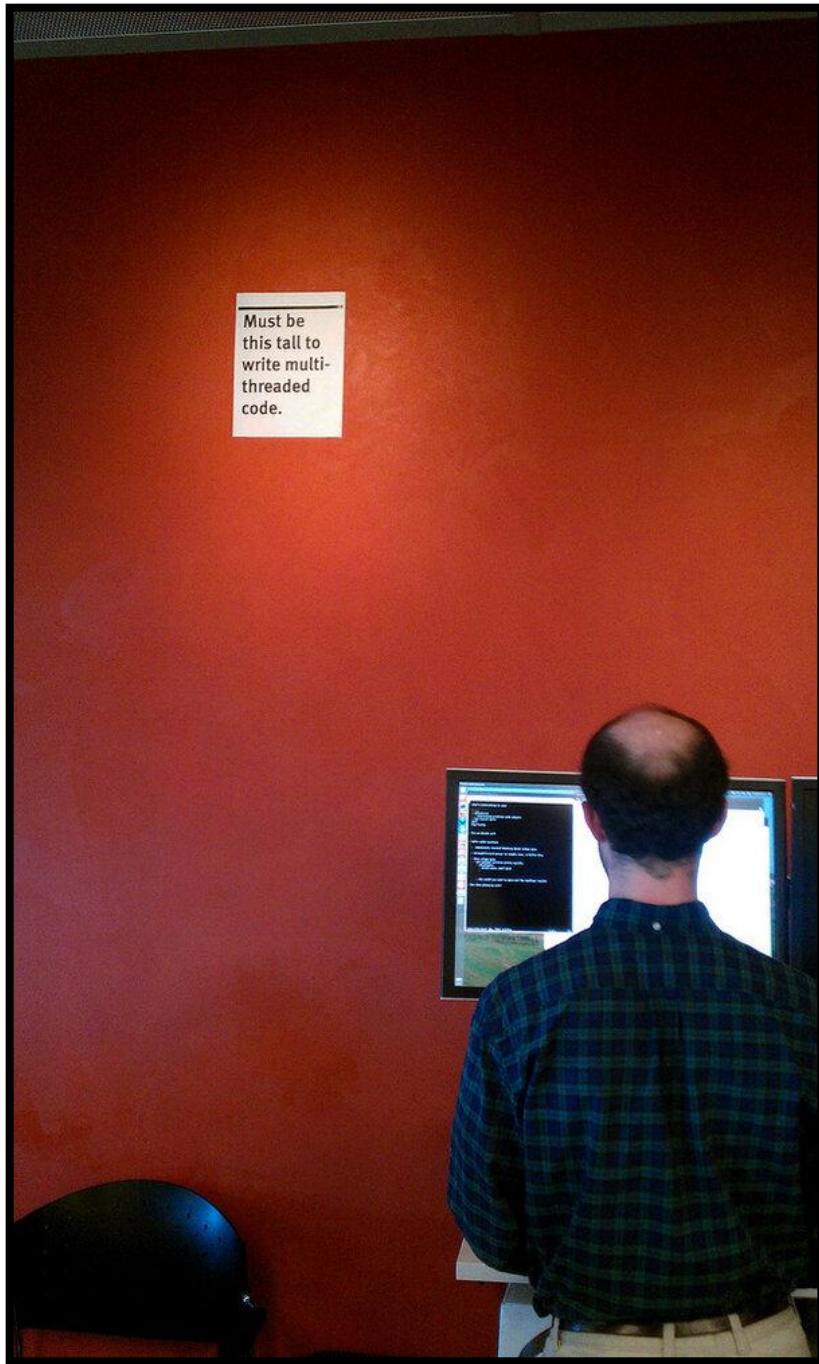
Andando avanti con la prenotazione dichiararsi di accettare le condizioni di cui sopra.

[Termini e Condizioni.](#)

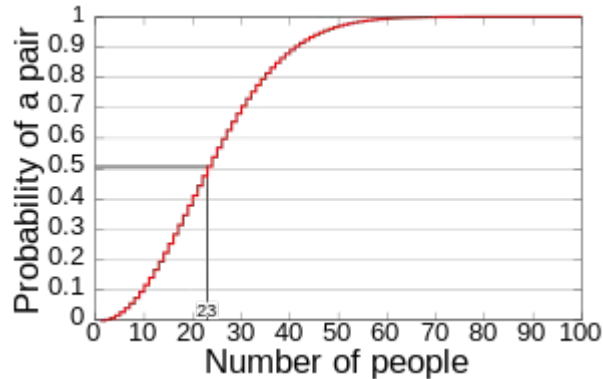


Ho detto che vi troverò





Paradosso del compleanno



- Sito web con N oggetti distinti disponibili
- Utenti online in contemporanea necessari ad avere Prob di conflitto $>50\% = \text{SQRT}(N)$
- Esempio: 365 oggetti □ con 23 persone $>50\%$

Master of race conditions: Costrutti di sincronizzazione storici

- Spinlock
- Test & Set
- Monitor
- Semafori
- **Lock**

Lock e blocchi synchronized

1. Un lock può essere posseduto da un thread alla volta.
2. Ogni lock può essere *occupato* o *libero*.
3. `acquire()`: se il lock è libero, acquisisce il lock e lo marca come *occupato*. Se un thread T cerca di prendere il possesso di un lock già *occupato*, T viene posto in stato di Wait.
4. `release()`: quando un lock viene liberato, uno tra i thread in Wait sullo stesso lock viene svegliato e posto in stato di «ready» (prenderà *probabilmente* il possesso del lock); **non è garantito che l'ordine di risveglio sia FIFO a meno che l'implementazione del lock non sia esplicitamente di tipo *fair* (es. *FairLock* di Java).**
5. Lock *rientrante*: un thread che possiede un lock *rientrante* può riacquisirlo quante volte vuole senza bloccarsi (e cioè può invocare `acquire()` tante volte di fila).
6. Lock *non rientrante*: un thread che possiede un lock entra in ciclo di attesa infinito se prova ad acquisire un lock che già possiede (e cioè se invoca `acquire()` due volte di fila).
7. (Java) Ogni ISTANZA di classe possiede un UNICO lock nascosto usabile facendo uso di metodi *synchronized* o blocchi *synchronized*;

Condition: notify() e wait()

1. Per ogni lock L, esiste un insieme di thread in attesa di acquisire L (WAIT-L)
2. Per ogni condition C esiste un insieme di thread in attesa su tale condition (WAIT-C)
3. Ogni condition C ha un lock padre (uno solo);
4. *C.wait()*: libera il lock di appartenenza e pone il thread chiamante in stato di attesa su WAIT-C;
5. NON E' POSSIBILE chiamare wait() se non si possiede il lock corrispondente;
6. *C.notify()*: prende un thread scelto in maniera IMPREDICIBILE da WAIT-C e lo sposta in WAIT-L;
7. *C.notifyAll()*: prende tutti i thread presenti in WAIT-C e li sposta in WAIT-L

Blocking queues



Quest'opera è distribuita con Licenza
Creative Commons Attribuzione - Non commerciale 4.0 Internazionale.







[LONDON CITY AIRPORT \(LCY\)](#)

Cockpit view into London City Airport

128.427 visualizzazioni • 13 mag 2020

1660

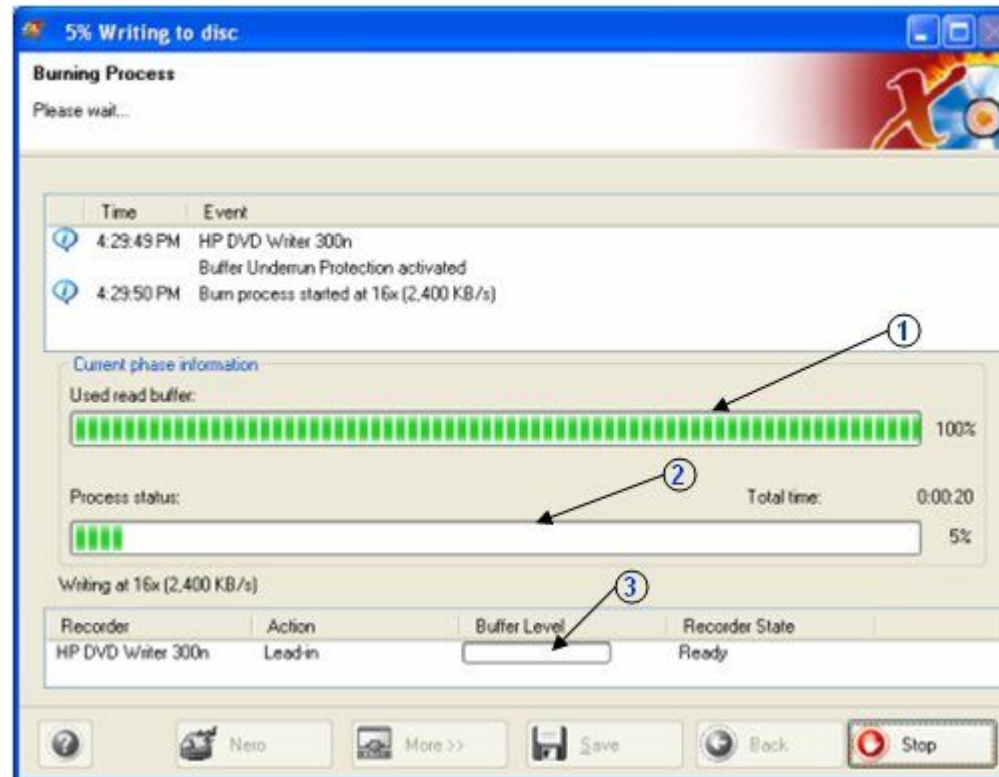
30

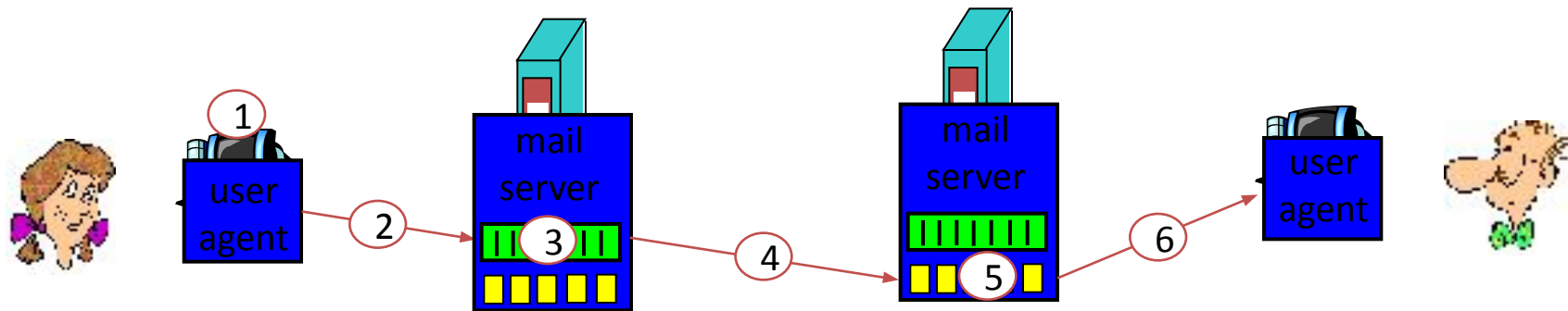
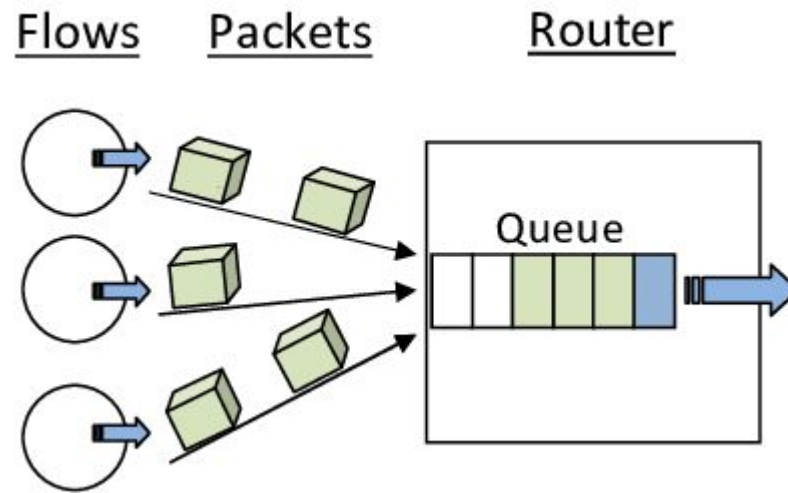
CONDIVIDI

SALVA

Riflessioni

- Il buffer della tastiera
- Il processo di masterizzazione di un DVD





Riflessioni - II

Esempi dove è utilissimo usare una coda bloccante:

- La comunicazione tra interlocutori in rete
- I mail server
- Le pizzerie

Code Bloccanti – Blocking Queue

- Uno strumento migliore di un coltellino svizzero
- Utile per
 - Mettere thread e/o processi in comunicazione
 - Distribuirsi compiti tra thread
 - Delegare compiti ad altri thread
 - Compensare velocità di elaborazione diverse

Idea

- Pensate al set di ordini per un pizzaiolo
- Ingredienti:
 - 1 Buffer FIFO
 - Metodi pensati per rendere *Thread-safe* l'accesso a quest'ultimo

- Quando un thread inserisce elementi in una Queue sta giocando il ruolo di **Produttore**
- Quando un thread preleva elementi da una Queue sta giocando il ruolo di **Consumatore**

Deadlock



Quest'opera è distribuita con Licenza
Creative Commons Attribuzione - Non commerciale 4.0 Internazionale.

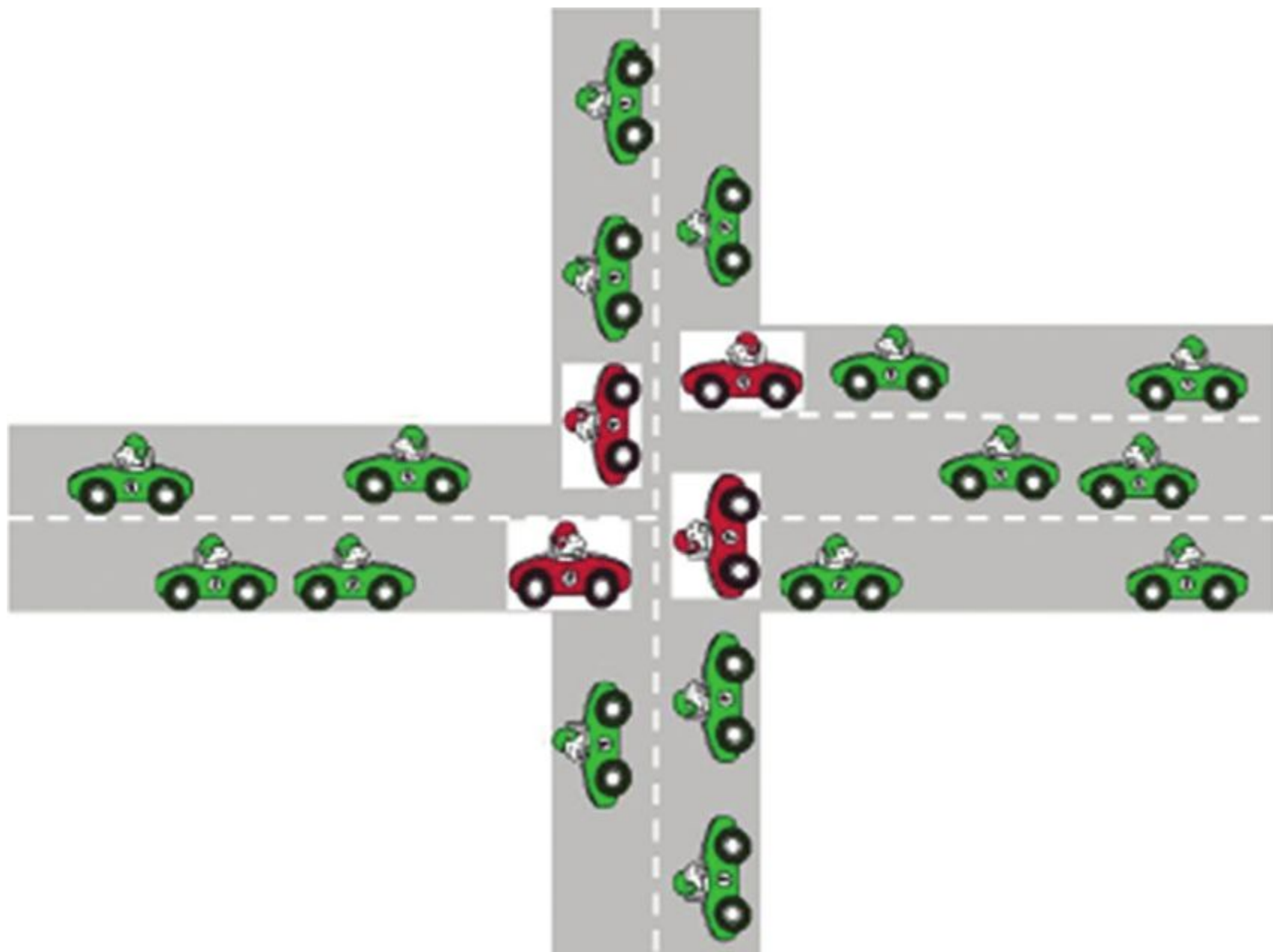


Deadlock

- Possibile quando si bloccano più risorse in contemporanea, formando *un ciclo* nel grafo di attesa
- Altre condizioni di deadlock più subdole sono possibili (es. usando le BlockingQueue)

GO AND POOP,
SO **WE** CAN LEAVE.

LEAVE, SO **I**
CAN GO AND POOP.



Evitare il Deadlock

1. Se possibile, riprogettare l'accesso alle risorse causa del ciclo con una disciplina diversa
 - es. mettere lock sulle risorse solo se tutte libere oppure aspettare
2. Prendere il lock alle risorse sempre nello stesso ordine
3. Prevenire... evitare di curare

Nested Lockout

```
public class Lockout {
```

```
    Lock a = new ReentrantLock();
    Lock b = new ReentrantLock();
    Condition c = b.newCondition();
    // T1:
    public void lock()
    {
        a.lock();
        b.lock();
        while(qualcheCondizione)
            c.await();
        qualcheCondizione = true;
        b.unlock();
        a.unlock();
    }
```

```
// T2:
```

```
public void unlock()
{
    a.lock();
    b.lock();
    qualcheCondizione = false;
    c.signalAll();
    b.unlock();
    a.unlock();
}
```

Starvation

- Starvation = possibilità che il tempo di attesa prima di accedere a una determinata risorsa sia anche infinito
- In altre parole:

$$\nexists T | \forall t > T, P(t) = 0$$

Barriera

- Consente a più thread di aspettarsi l'un l'altro
- Utile per sincronizzare un gruppo di processi che eseguono parti di un compito frazionabile **in parti uguali**
- Implementato in `CyclicBarrier` di Java
- Implementato in `Barrier` di Python
- Un metodo: `await()` – Java, `wait()` - Python