

PKaya Operating System

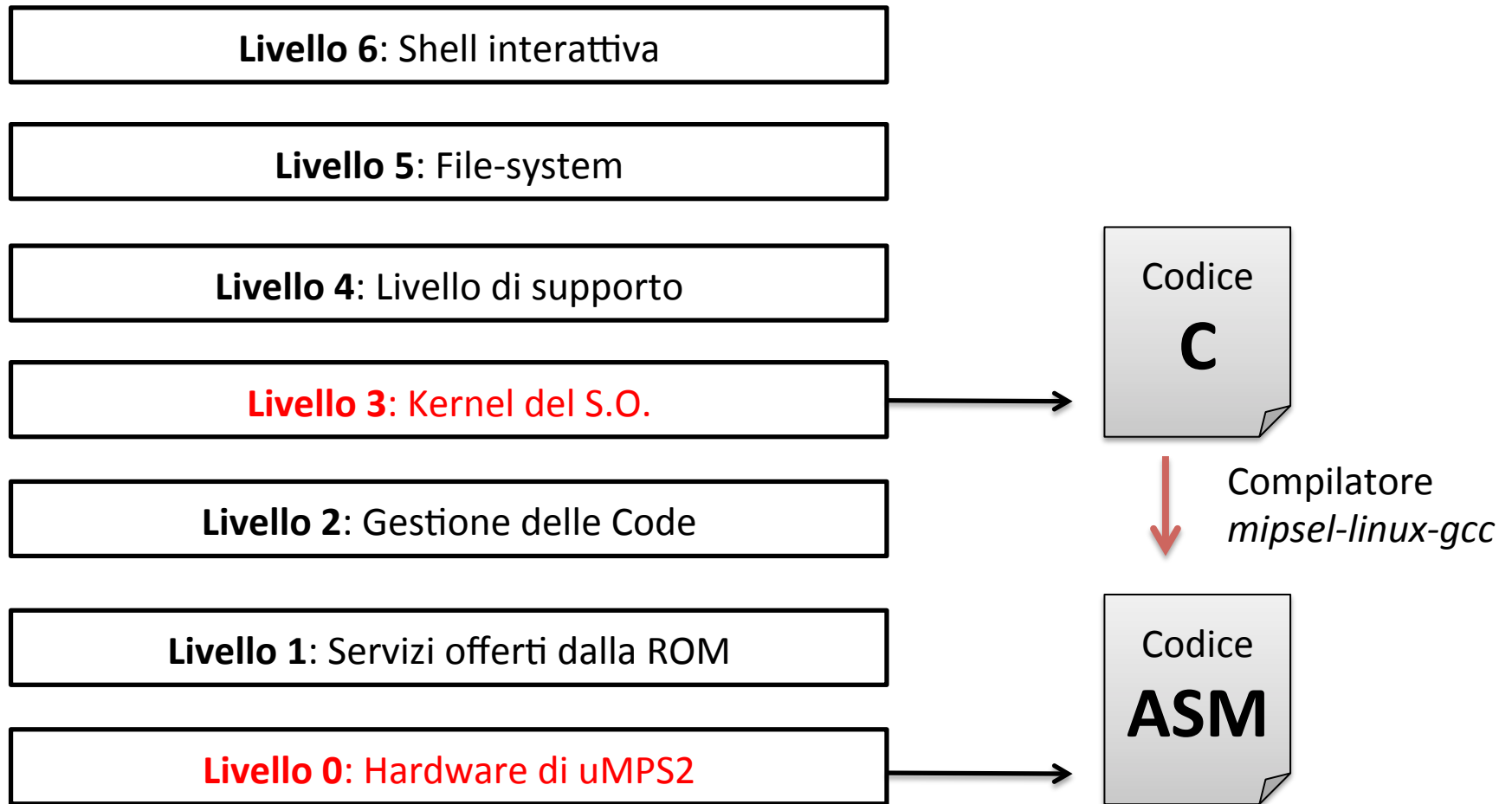
Librerie di supporto: LibUMPS

FASE 2

Anno Accademico 2011-2012

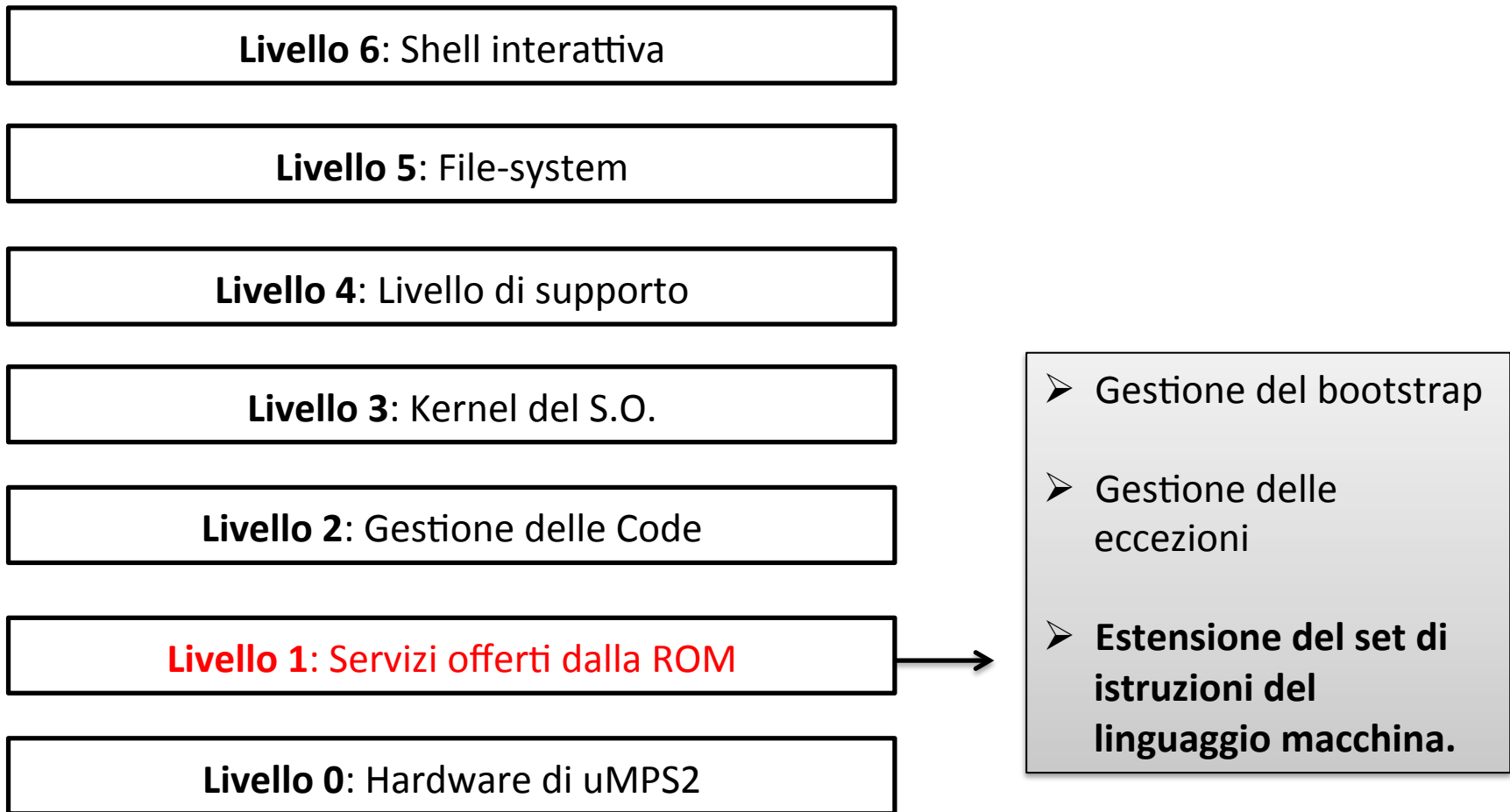
pKaya OS

- Sistema Operativo in 6 **livelli** di astrazione.



pKaya OS

- Sistema Operativo in 6 **livelli** di astrazione.



pKaya: File di Supporto

- **libumps**: librerie di supporto di UMPS2
- **Wrapper** per le istruzioni della ROM:
 - Eseguire **istruzioni** della ROM (tramite codice C)
 - Accedere ad i **registri** del co-processore **CP0**
 - Eseguire istruzioni **SYSCALL** e TLB
- libumps e' composto da due parti:
 - **libumps.h**: da **includere** nei sorgenti C
 - **libumps.o**: da **linkare** con gli altri file oggetto per produrre l'eseguibile.

pKaya: File di Supporto

- **const.h**: File contenente **costanti** e **macro** utili per lo sviluppo di Fase2

Esempi:

- Costanti: Indirizzi di memoria
- Costanti: Linee di interrupt
- Costanti: Maschere di bit
- Costanti: Id delle Syscall
- Macro: Gestione dei timer
- Macro: Calcolo del device register address
-

pKaya: File di Supporto

- **uMPStypes.h**: File contenente **strutture dati** utili per lo sviluppo di Fase2

Esempi:

- **state_t**: stato di un processore (CPU)

```
typedef struct {  
    U32 entry_hi;  
    U32 cause;  
    U32 status;  
    U32 pc_epc; /* pc in the new area, epc in the old area */  
    U32 gpr[29];  
    U32 hi;  
    U32 lo;  
} state_t;
```

pKaya: Libreria libumps

Istruzioni della **ROM**:

- *Estensioni* del linguaggio assembly ...
- Alcune eseguibili solo in **Kernel Mode**
- Utilizzano indirizzi **fisici** (non virtuali!)
- Invoke mediante istruzioni **BREAK**
- Scritte in Assembly MIPS, ma **wrapper C** attraverso **libumps** ...

pKaya: Libreria **libumps**

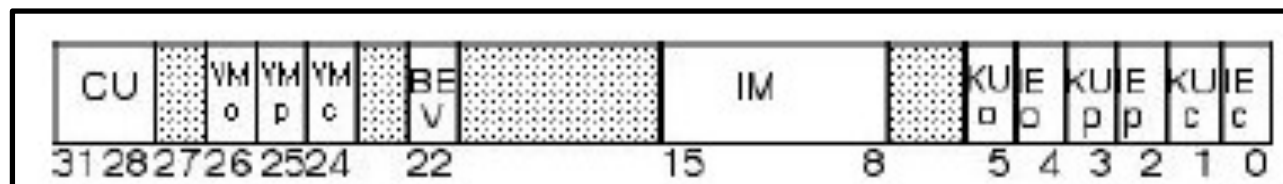
➤ **Stato** della CPU/sistema:

Metodo C	Istruzione ROM	Descrizione
void HALT ()	HALT	<i>Spegne la macchina</i>
void PANIC ()	PANIC	<i>Kernel panic!</i>
Void WAIT ()	WAIT	<i>Mette la CPU in stato idle</i>

pKaya: Libreria **libumps**

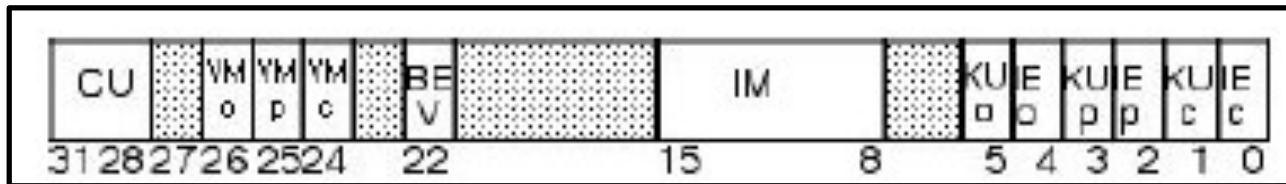
➤ Accesso ai **registri** del **CP0**:

Metodo C	Registro	Descrizione
unsigned int setStatus (unsigned int)	Status	<i>Setta il registro Status</i>
unsigned int getStatus()	Status	<i>Restituisce il valore del registro</i>



pKaya: Libreria **libumps**

➤ Accesso ai **registri** del **CP0**:



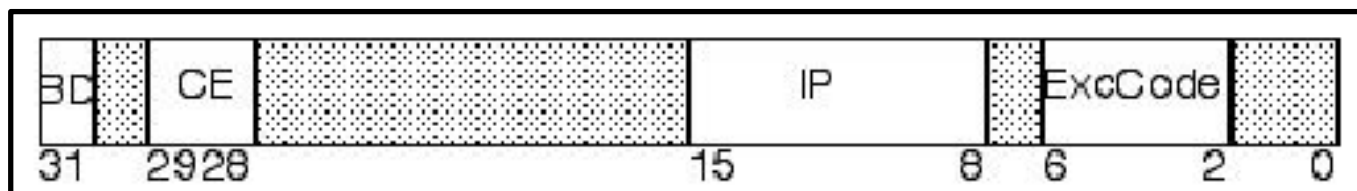
Es. Settare Kernel Mode ON agendo sul registro CP0:

```
...
status=getSTATUS();
status |= (STATUS_KUc); // maschera di bit definita in const.h
setSTATUS(status);
....
```

pKaya: Libreria **libumps**

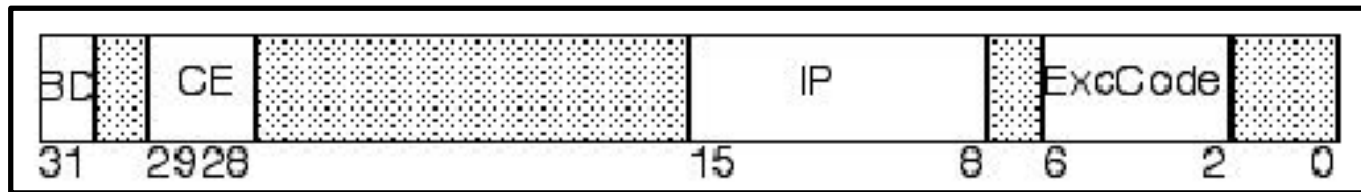
➤ Accesso ai **registri** del **CP0**:

Metodo C	Registro	Descrizione
unsigned int setCAUSE (unsigned int)	Cause	<i>Setta il registro Cause</i>
unsigned int getCAUSE()	Cause	<i>Restituisce il valore del registro</i>



pKaya: Libreria **libumps**

➤ Accesso ai **registri** del **CP0**:



Es. Verificare quale linea ha causato un interrupt ...

```
...  
int cause=getCAUSE();  
  
    if (CAUSE_IP_GET(cause, INT_TIMER)) { // Macro in const.h  
...  
....
```

pKaya: Libreria **libumps**

➤ Operazioni con **Timer**:

Metodo C	Registro	Descrizione
unsigned int setTIMER (unsigned int)	Timer	<i>Setta il timer della CPU</i>
unsigned int getTIMER()	Timer	<i>Preleva il valore del timer</i>

- Operazioni con Timer del BUS (in const.h, non libumps):
- void **setIT**(unsigned int)
 - unsigned int **getGET_TODLOW**()

pKaya: Libreria **libumps**

- **Esempio1:** fare in modo che ogni 5 secondi venga incrementata una variabile, inizializzata a 0. Quando la variabile diventa uguale a 5, spegnere la macchina ...

pKaya: Libreria **libumps**

```
int main(void){

    state_t* new_area = (state_t *) INT_NEWAREA;
    /* interrupt disabilitati, kernel mode e memoria virtuale spenta */
    new_area->pc_epc = new_area->reg_t9 = (memaddr)timerHandler;
    new_area->status &= ~(STATUS_IEc | STATUS_KUc | STATUS_VMc);
    new_area->reg_sp = RAMTOP;

    int status=0;
    status |= (STATUS_IEc | STATUS_INT_UNMASKED);
    setSTATUS(status);

    SET_IT(5000000);

    while(1) {}
}
```

pKaya: Libreria **libumps**

```
void timerHandler() {  
  
    int cause=getCAUSE();  
  
    if (CAUSE_IP_GET(cause, INT_TIMER)) {  
  
        numTimes++;  
  
        if (numTimes >5)  
            HALT();  
        SET_IT(50000000);  
  
    }  
}
```


pKaya: Libreria **libumps**

➤ Altre operazioni sui **registri** CP0:

unsigned int **getINDEX()** → registro CP0: Index

unsigned int **getENTRYHI()** → registro CP0: EntryHi

unsigned int **getENTRYLO()** → registro CP0: EntryLo

unsigned int **getPRID()** → registro CP0: PRID

unsigned int **getRANDOM()** → registro CP0: Random

unsigned int **getEPC()** → registro CP0: EPC

unsigned int **getBADVADDR()** → registro CP0: BadVAddr

unsigned int **setINDEX(unsigned int)** → registro CP0: Index

unsigned int **setENTRYHI(unsigned int)** → registro CP0: EntryHi

unsigned int **setENTRYLO(unsigned int)** → registro CP0: EntryLo

pKaya: Libreria **libumps**

➤ Operazioni sui registri **TLB**:

void **TLBWR**() → Istruzione ROM: TLB-Write-Random

void **TLBWI**() → Istruzione ROM:TLB-Write-Index

void **TLBR**() → Istruzione ROM:TLB-Read

void **TLBP**() → Istruzione ROM:TLB-Probe

void **TLBCLR**() → Istruzione ROM: TLB-Clear

pKaya: Libreria **libumps**

➤ Load/Store dello **Stato della CPU**:

Metodo C	Istruzione	Descrizione
unsigned int STST (state_t * statep)	STST	<i>Salva lo stato della CPU in memoria</i>
unsigned int LDST (state_t * statep)	LDST	<i>Carica lo stato della CPU da *statep (atomica)</i>
unsigned int FORK (unsigned int entryhi, unsigned int status, unsigned int pc, state_t * statep)	FORK	<i>Carica lo stato della CPU da *statep (NON atomica)</i>

pKaya: Libreria **libumps**

- **Esempio2:** caricamento nella CPU di uno *state_t* (associato ad un processo) ...

pKaya: Libreria **libumps**

```
void pcode() {
```

```
    int i;  
    for (i=0; i<30000000; i++);
```

```
    HALT();  
}
```

```
int main(void){
```

```
    state_t pstate;  
    STST(&pstate);  
    pstate.reg_sp = RAMTOP;  
    pstate.pc_epc = pstate.reg_t9 = (memaddr)pcode;  
    pstate.status = pstate.status | STATUS_IEp | STATUS_INT_UNMASKED;
```

```
    LDST(&pstate);  
    PANIC();  
}
```

pKaya: Libreria **libumps**

➤ Avvio di una **CPU** (uMPS2):

Metodo C	Istruzione	Descrizione
<pre>unsigned int void INITCPU(uint32_t cpuid, state_t *start_state, state_t *state_areas);</pre>	INITCPU	<i>Invia un segnale di RESET alla CPU con id pari a cpuid. Lo stato della CPU viene inizializzato a *start_state Le new/old area della CPU sono in *state_areas</i>

pKaya: Libreria **libumps**

- **Esempio3:** avvio di 2 processi concorrenti su CPU0 e CPU1. Il primo processo (su CPU0) incrementa una variabile, il secondo processo (su CPU1) spegne la macchina quando la variabile raggiunge un valore soglia ...

pKaya: Libreria **libumps**

```
#define MAX_CPUS 2
```

```
int main(void){
```

```
    state_t new_old_areas[MAX_CPUS][8];
```

```
    state_t pstate, pstate2;
```

```
    STST(&pstate);
```

```
    pstate.reg_sp = RAMTOP;
```

```
    pstate.pc_epc = pstate.reg_t9 = (memaddr)pcode2;
```

```
    pstate.status = pstate.status | STATUS_IEp | STATUS_INT_UNMASKED;
```

```
    INITCPU(1,&pstate,&new_old_areas[1]);
```

```
    STST(&pstate2);
```

```
    pstate2.reg_sp = RAMTOP - QTABLE;
```

```
    pstate2.pc_epc = pstate2.reg_t9 = (memaddr)pcode;
```

```
    pstate2.status = pstate2.status | STATUS_IEp | STATUS_INT_UNMASKED;
```

```
    LDST(&pstate2);
```

```
}
```


pKaya: Libreria **libumps**

```
int end=0;
```

```
// Process 1
```

```
void pcode() {
```

```
    long i=0;
```

```
    for (i=0; i<30000000; i++);
```

```
    end=1;
```

```
}
```

```
// Process 2
```

```
void pcode2() {
```

```
    while (!end);
```


```
    HALT();
```

```
}
```

pKaya: Libreria **libumps**

➤ Implementazione **sezioni critiche**

Metodo C	Istruzione	Descrizione
int CAS (uint32 t *area, uint32 t ov, uint32 t nv)	CAS (atomica)	<i>Setta l'area puntata da *area con il valore di nv, se il valore attuale di *area e' pari a ov. Restituisce 1 in caso di update, 0 altrimenti.</i>



```
atomic {  
    if (*area == ov) {  
        *area=nv;  
        return 1;  
    }  
    else return 0;  
}
```