SAPIENZA
UNIVERSITÀ DI ROMA

Sistemi di Controllo Real Time

Automazione

Vincenzo Suraci

STRUTTURA DEL NUCLEO TEMATICO

- SISTEMI REAL TIME
- CLASSIFICAZIONE DEI SISTEMI REAL TIME
- PARALLELISMO E PROGRAMMAZIONE CONCORRENTE

SISTEMI REAL TIME

SISTEMI REAL TIME

Un sistema di controllo che si propone lo scopo di coordinare un insieme definito di elementi singoli deve necessariamente pianificare un **ALGORITMO** per **SEQUENZIALIZZARE** e **TEMPORIZZARE** gli interventi degli elementi singoli per raggiungere un ben preciso obiettivo di produzione.

Le **TECNOLOGIE** utilizzate per implementare l'algoritmo di controllo al livello di coordinamento sono, per la maggioranza dei sistemi di automazione industriale attuali, basate su **PROGRAMMAZIONE SOFTWARE** di opportuni **MICROPROCESSORI**.

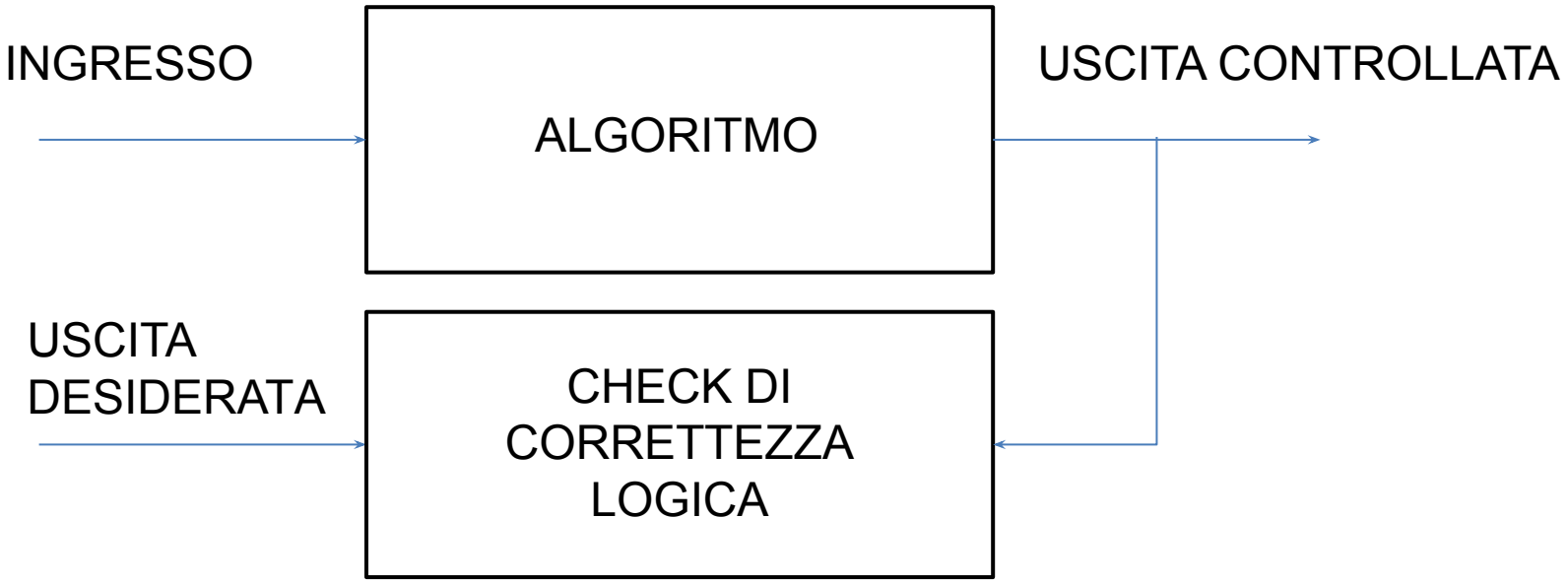
Per un corretto funzionamento di un insieme di elementi singoli è fondamentale che il sistema di controllo monitori costantemente lo stato dell'impianto e impartisca i **COMANDI ADEGUATI** ogni qualvolta vengano determinate opportune condizioni.

Tali comandi devono essere eseguiti in un **TEMPO OPPORTUNO** affinché abbiano l'efficacia necessaria al raggiungimento degli obiettivi desiderati.

SISTEMI REAL TIME

L'**ALGORITMO** che implementa la legge di controllo si dice:

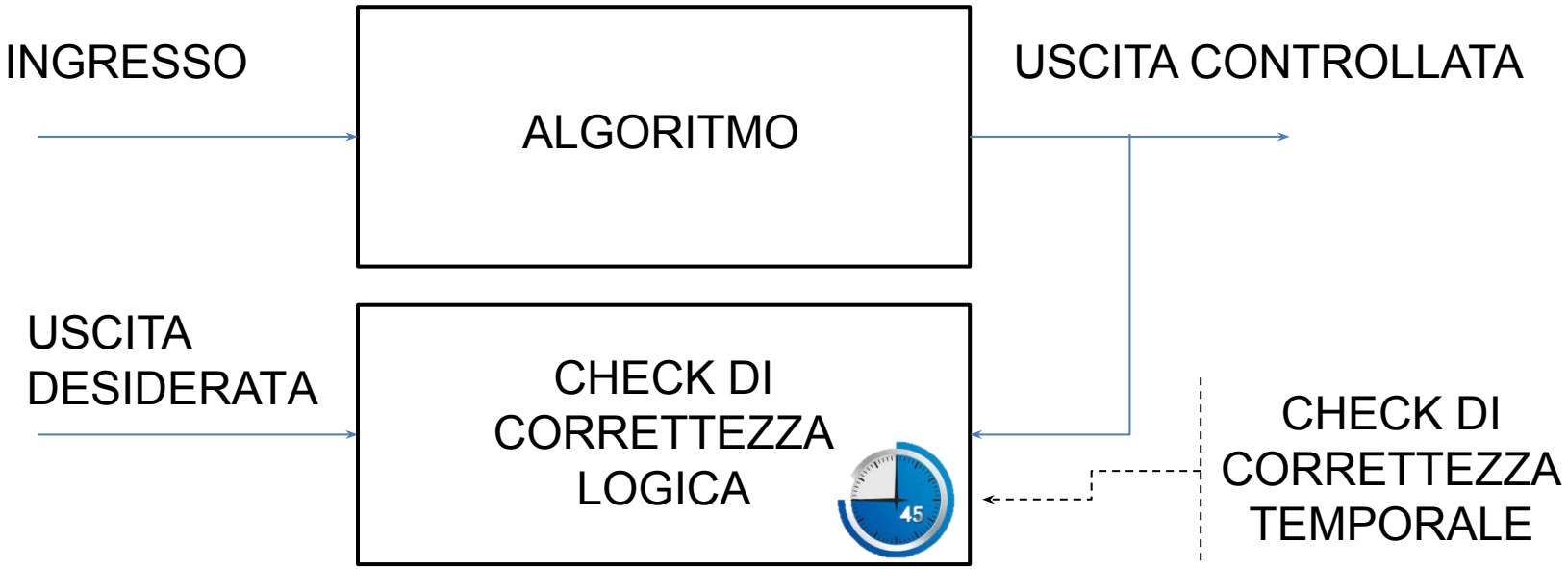
- **LOGICAMENTE CORRETTO** quando i risultati forniti sono quelli attesi a partire da ben definiti dati di ingresso;



SISTEMI REAL TIME

L'**ALGORITMO** che implementa la legge di controllo si dice:

- **LOGICAMENTE CORRETTO** quando i risultati forniti sono quelli attesi a partire da ben definiti dati di ingresso;
- **TEMPORALMENTE CORRETTO** quando i risultati sono forniti rispettando delle prestabilite specifiche temporali dette **DEADLINE**.



SISTEMI REAL TIME

Il SISTEMA DI CONTROLLO che elabora l'algoritmo, si può definire **SISTEMA REAL TIME** se e solo se:

1. è in grado di elaborare le informazioni in modo da fornire delle risposte **LOGICAMENTE CORRETTE**;

2. è in grado di elaborare le informazioni in modo da fornire delle risposte **TEMPORALMENTE CORRETTE**;

SISTEMI REAL TIME

Una considerazione intuitiva porta erroneamente a classificare come SISTEMI REAL TIME, quei sistemi che hanno **CAPACITÀ COMPUTAZIONALI** estremamente **ELEVATE**.

In realtà non è né sufficiente, né necessario avere un **MICROPROCESSORE** estremamente potente per garantire che esso costituisca un **SISTEMA REAL TIME**.

- La **correttezza logica** dipende da come è stato implementato il programma di controllo. Un **BUG** nel software mina tale requisito, indipendentemente dalla CPU;
- La **correttezza temporale** dipende da molti fattori HW e SW: dal BIOS, dal OS, dalla modalità di gestione degli IRQ, dalla gestione del BUS, dalla gestione delle periferiche, etc.

SISTEMI REAL TIME

Il soddisfacimento dei requisiti di correttezza logica e temporale devono essere soddisfatti nella **TOTALITÀ DEI CASI** e **NON MEDIAMENTE** durante tutto il ciclo di vita del sistema di controllo.

Pertanto il comportamento di un sistema real time deve essere **PREVEDIBILE**.

La proprietà di un sistema di controllo di avere un comportamento prevedibile è nota anche come **COMPORTAMENTO DETERMINISTICO**.

Per ottenere sistemi di controllo REAL TIME è pertanto necessario studiare, progettare ed adottare **ARCHITETTURE HARDWARE E SOFTWARE DEDICATE** che rispettino tale proprietà.

CLASSIFICAZIONE DEI SISTEMI REAL TIME

SISTEMI REAL TIME

Dato un impianto industriale, non tutti i dispositivi devono essere necessariamente governati da un sistema di controllo real time.

Nel caso di un **ascensore**, ad esempio, è importante avere un controllo real time della parte legata alla **sicurezza fisica delle persone** (rilevazione apertura e chiusura delle porte, posizione e velocità dell'abitacolo, ecc.).

Non è necessario avere un controllo real time della parte legata alla **prenotazione della chiamata**.

SISTEMI REAL TIME

In tal senso è utile definire il concetto di **Quality of Service (QoS)** di un sistema di controllo esprimibile in funzione della capacità di rispettare i vincoli di correttezza logica e temporale.

Ad esempio si possono definire due parametri di QoS in termini di:

- ***coefficiente di correttezza logica*** pari al rapporto tra

$$\frac{\text{RISULTATI CORRETTI}}{\text{TOTALE DEI RISULTATI ELABORATI}}$$

- ***coefficiente di correttezza temporale*** pari al rapporto

$$\frac{\text{RISULTATI ELABORATI ENTRO LA DEADLINE}}{\text{TOTALE DEI RISULTATI ELABORATI}}$$

SISTEMI HARD REAL TIME

Un sistema di controllo si dice HARD REAL TIME se e solo se:

- **NON VIOLA MAI IL VINCOLO DI CORRETTEZZA LOGICA** (i task danno sempre risultati corretti)
- **NON VIOLA MAI IL VINCOLO DI CORRETTEZZA TEMPORALE** (i task non superano mai le deadline)

Ovverosia, un sistema di controllo si dice HARD REAL TIME se e solo se:

- *coefficiente di correttezza logica = 1*
- *coefficiente di correttezza temporale = 1*

Un sistema di controllo HARD REAL TIME viene richiesto ogni qualvolta anche solo una violazione dei vincoli di correttezza logica o temporale possono **compromettere l'incolumità di persone o cose, oppure la funzionalità stessa del sistema controllato** (si pensi, ad esempio, agli impianti chimici o alle centrali termonucleari).

SISTEMI SOFT REAL TIME

Ogni qualvolta non è necessario ricorrere ad un sistema HARD REAL TIME, si può optare per l'adozione di un sistema **SOFT REAL TIME**.

Un sistema di controllo si dice **SOFT REAL TIME** se e solo se:

- può violare il vincolo di correttezza logica (i task possono dare risultati scorretti)
- può violare il vincolo di correttezza temporale (i task possono superare le deadline)

Ovverosia, un sistema di controllo si dice SOFT REAL TIME se:

Min (coefficiente di correttezza logica, coefficiente di correttezza temporale) < 1

Differenti sistemi di controllo SOFT REAL TIME possono pertanto essere classificati in base alla QoS che sono in grado di offrire.

VINCOLI REAL TIME

Esiste una relazione tra i requisiti dei sistemi di controllo real time e la velocità di esecuzione dei task da parte del sistema controllato.

Tale relazione è espressa come **RAPPORTO** tra il TEMPO IMPIEGATO dal sistema ad eseguire il task e la DEADLINE. Tale rapporto è detto **VINCOLO REAL TIME**.

$$\textbf{VINCOLO REAL TIME} = \frac{\text{TEMPO DI ESECUZIONE DEL TASK}}{\text{DEADLINE}}$$

VINCOLI REAL TIME

- Un VINCOLO REAL TIME si dice **LARGO** se le deadline temporali sono ampie rispetto al tempo necessario a svolgere le operazioni previste.

- Un VINCOLO REAL TIME si dice **STRETTO** se le deadline temporali sono prossime al tempo di calcolo necessario a svolgere le operazioni previste.

Realizzare un sistema HARD REAL TIME con VINCOLI REAL TIME STRETTI implica una progettazione sistematica delle architetture hardware e software per sfruttare al meglio le risorse a disposizione.

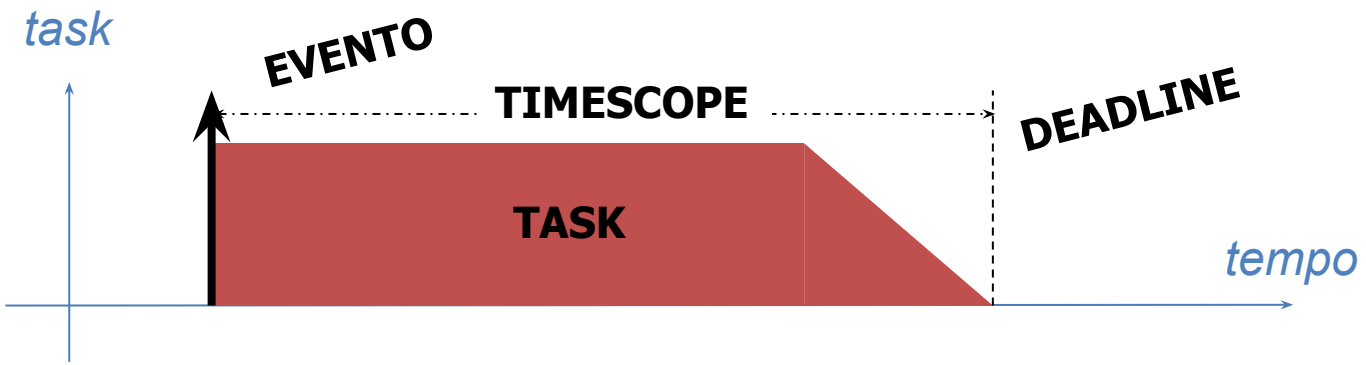
PARALLELISMO E PROGRAMMAZIONE CONCORRENTE

DEFINIZIONE DI TASK

Un sistema di controllo real time deve spesso coordinare attività tra loro completamente differenti e scorrelate. Pertanto è bene definire le attività in maniera indipendente da qualsiasi caratterizzazione tecnologica.

DEFINIZIONE

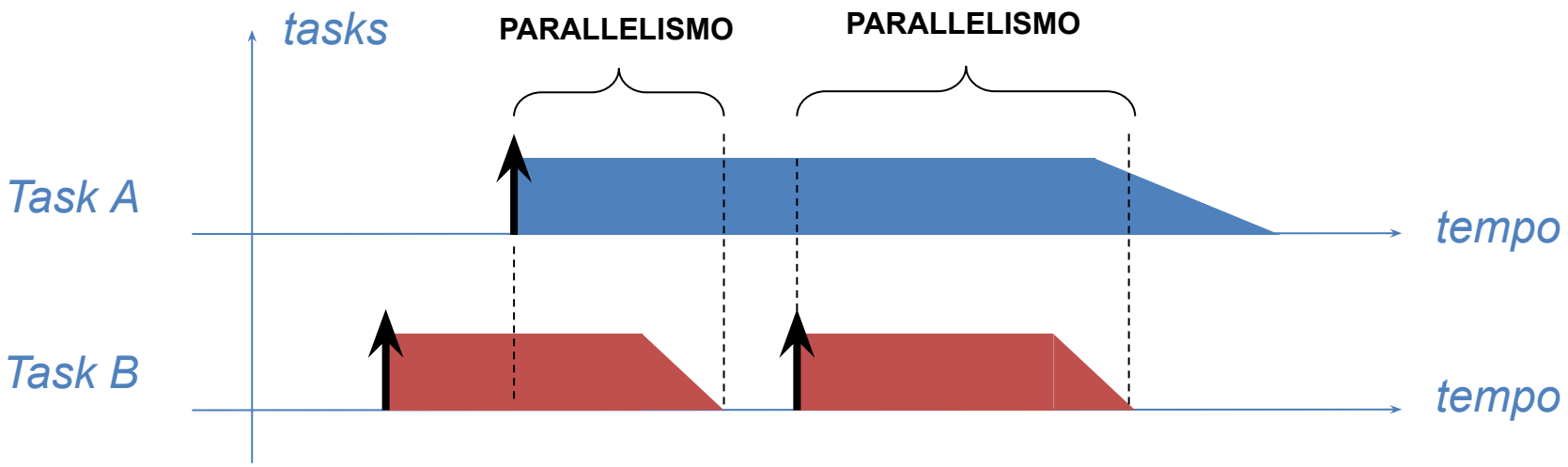
Dicesi **TASK** (o PROCESSO) una **unità atomica di lavoro**, innescata da un **EVENTO**, che deve terminare entro un istante di tempo chiamato **DEADLINE**. L'intervallo di tempo tra l'evento e la deadline è detto **TIMESCOPE** del task.



PARALLELISMO

Il sistema di controllo real time deve spesso coordinare diversi elementi singoli che evolvono indipendentemente l'uno dall'altro.

L'indipendenza dei Task associati ad elementi diversi, genera immediatamente un problema di coordinamento dovuto alla necessità di svolgere CONTEMPORANEAMENTE più di un Task in parallelo. Questo problema è noto come **PARALLELISMO**.



PARALLELISMO

Per poter svolgere contemporaneamente le attività previste da Task paralleli, la architettura del sistema di controllo deve essere accuratamente progettata.

- Si parla di architettura **MONOPROCESSORE** quando una sola unità di calcolo, che è in grado di eseguire le attività di un solo task per volta, deve eseguire i compiti richiesti dai diversi task.

- Si parla di architettura **MULTIPROCESSORE** quando più unità di calcolo possono eseguire contemporaneamente i compiti richiesti dai diversi task;

PARALLELISMO

- Quando il numero massimo di Task paralleli è maggiore del numero di unità di calcolo disponibili nel sistema di controllo si deve attuare una strategia di **PARALLELISMO LOGICO.**

- Quando il numero massimo di Task paralleli è minore o uguale al numero di unità di calcolo disponibili nel sistema di controllo, si può attuare una strategia di **PARALLELISMO REALE**.

È bene sottolineare che anche sistemi di controllo con architetture MULTIPROCESSORE possono applicare strategie di PARALLELISMO LOGICO, per poter eseguire in maniera parallela (ma non contemporanea) anche un numero di Task maggiore del numero di unità di calcolo disponibili.

PROGRAMMAZIONE CONCORRENTE (SCHEDULING)

Quando una stessa unità di calcolo, o PROCESSORE, deve eseguire più Task in parallelo, deve applicare una strategia di gestione che sequenzializzi l'uso dell'unica risorsa fisica disponibile (il processore) in modo da terminare le attività previste per i differenti Task entro le relative deadline.

Lo studio e la definizione di questa strategia di gestione è chiamato **PROGRAMMAZIONE CONCORRENTE** o **SCHEDULING.**

PROGRAMMAZIONE CONCORRENTE (SCHEDULING)

Nella maggior parte dei sistemi di automazione di complessità media è sempre verificato che il numero di Task paralleli sia maggiore del numero delle unità di calcolo presenti nel sistema di controllo.

Spesso infatti i sistemi di controllo sono **MONOPROCESSORE** e richiedono, pertanto, la gestione via HARDWARE o via SOFTWARE di un ALGORITMO DI SCHEDULING.

Senza perdita di generalità, nel seguito analizzeremo spesso il caso di sistemi di controllo MONOPROCESSORE in quanto sono i sistemi più diffusi nel settore della Automazione.

STATI DI UN TASK

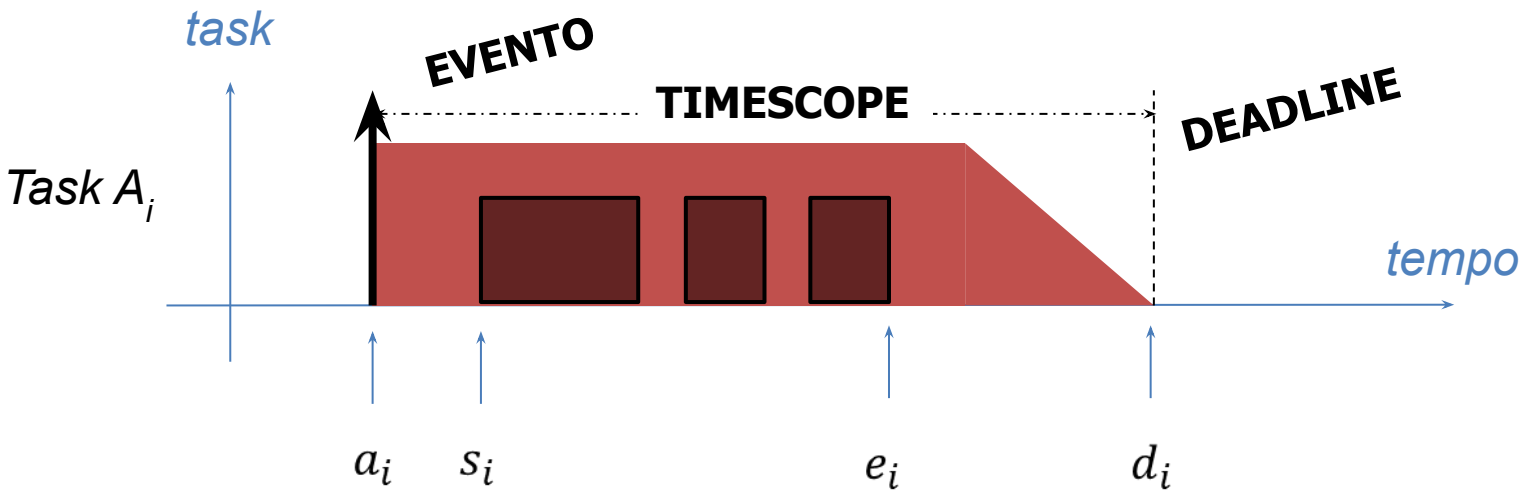
Dato un istante di tempo t, ed un generico Task, possiamo stabilire che esso è:

- **ATTIVO** se l'evento che lo ha innescato è avvenuto prima dell'istante t e la deadline scadrà dopo l'instante t;
- **PRONTO** se il task è attivo, ma non è eseguito dal processore nell'istante di tempo t;
- **IN ESECUZIONE** se il task è attivo ed è eseguito dal processore nell'istante di tempo t;

PARAMETRI CARATTERISTICI DI UN TASK

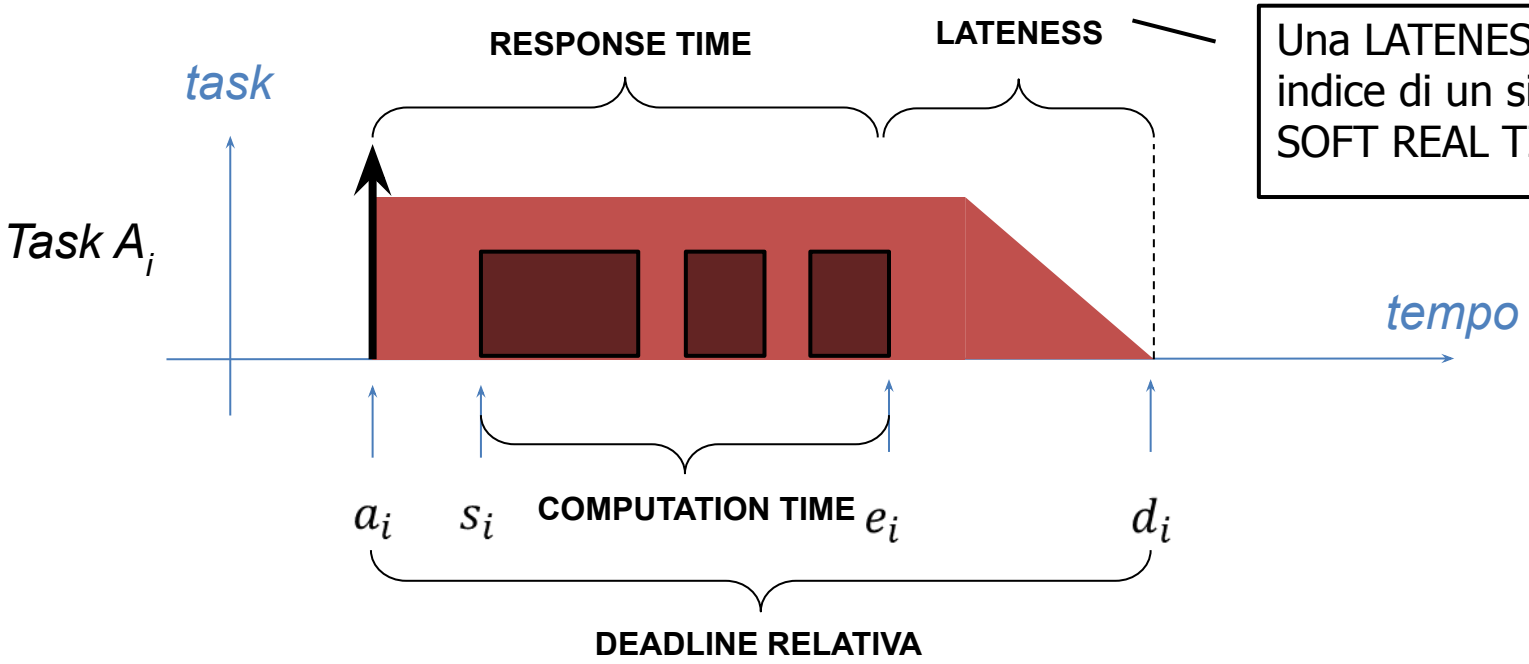
Dato un task A_i si possono definire i seguenti **PARAMETRI CARATTERISTICI**:

- a_i – **ACTIVATION TIME** – istante in cui il task diventa ATTIVO
- s_i – **START TIME** – istante in cui il task viene ESEGUITO per la prima volta
- e_i – **END TIME** – istante in cui l'esecuzione del task viene terminata definitivamente
- d_i – **DEADLINE ASSOLUTA** – istante in cui il task deve essere completato



PARAMETRI CARATTERISTICI DI UN TASK

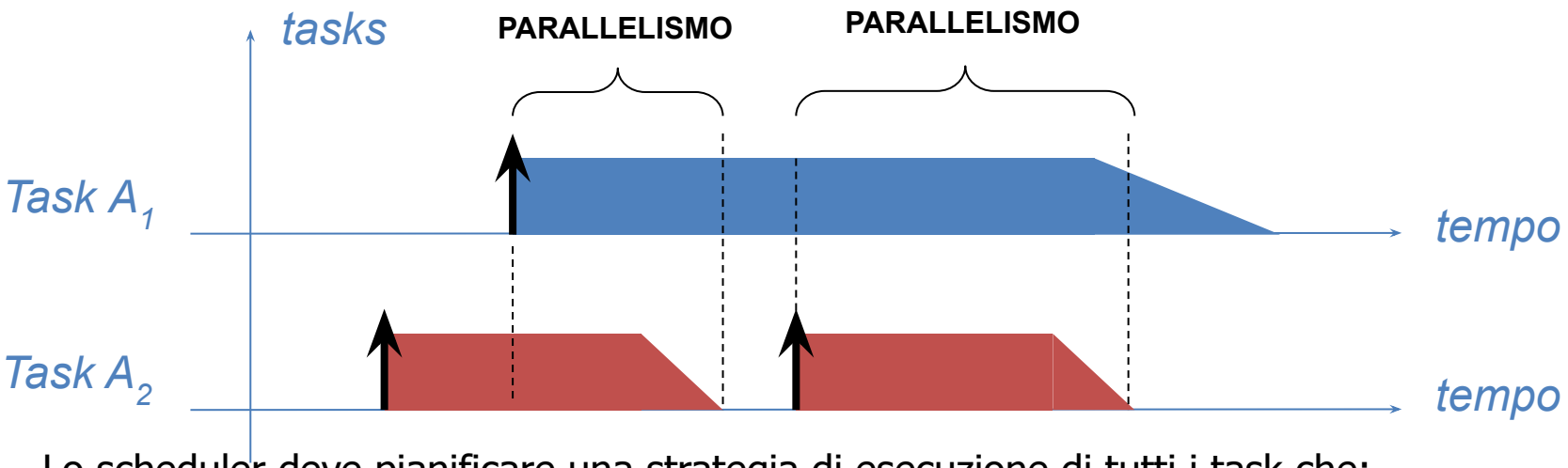
- $C_i = e_i - s_i$ **COMPUTATION TIME** – tempo complessivo di esecuzione del task
- $D_i = d_i - a_i$ **DEADLINE RELATIVA** – tempo massimo di esecuzione del task
- $R_i = e_i - a_i$ **RESPONSE TIME** – tempo impiegato per terminare il task
- $L_i = e_i - d_i$ **LATENESS** – ritardo tra la deadline e la terminazione del task



Una LATENESS positiva è indice di un sistema SOFT REAL TIME

RUOLO DELLO SCHEDULING

In una situazione di parallelismo come mostrato in figura:



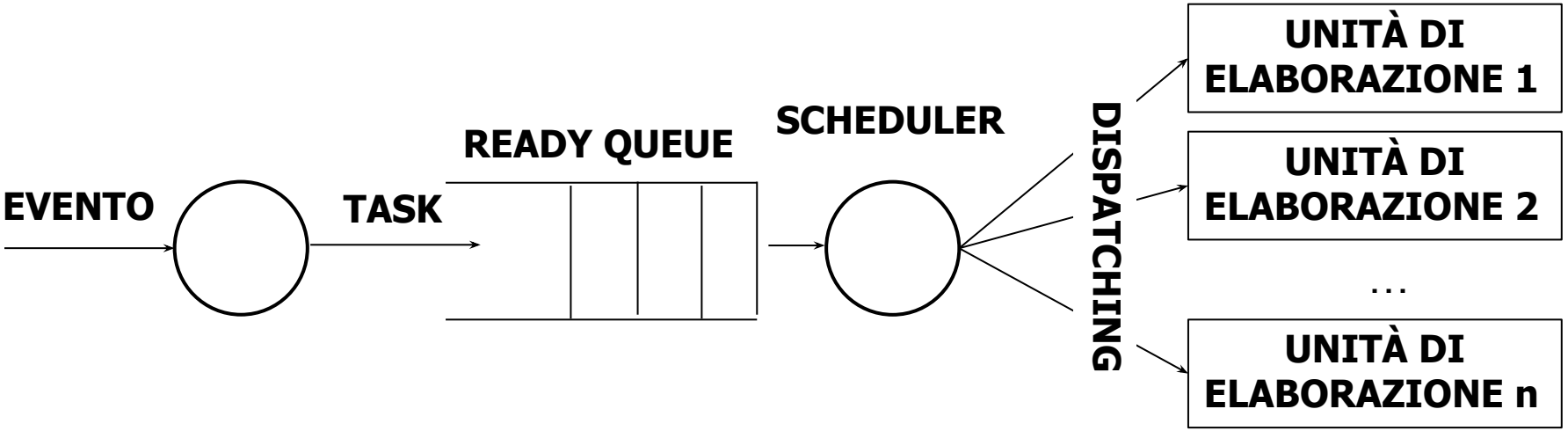
Lo scheduler deve pianificare una strategia di esecuzione di tutti i task che:

1. **Rispetti i vincoli** sulle risorse a disposizione (ad es. di calcolo),
2. **Rispetti le priorità** dei task (ad ed. quelli HARD real time da quelli SOFT real time)
3. **Massimizzi, quando possibile, opportuni indici prestazionali** (ad es. consumo energetico, occupazione di memoria, etc.)

RUOLO DELLO SCHEDULING

Quando si presentano situazioni di parallelismo, in cui nello stesso istante di tempo più task sono attivi, i task vengono (virtualmente) collocati in una coda di attesa (**READY QUEUE**). Quando un task viene terminato, esso può essere tolto dalla coda di attesa.

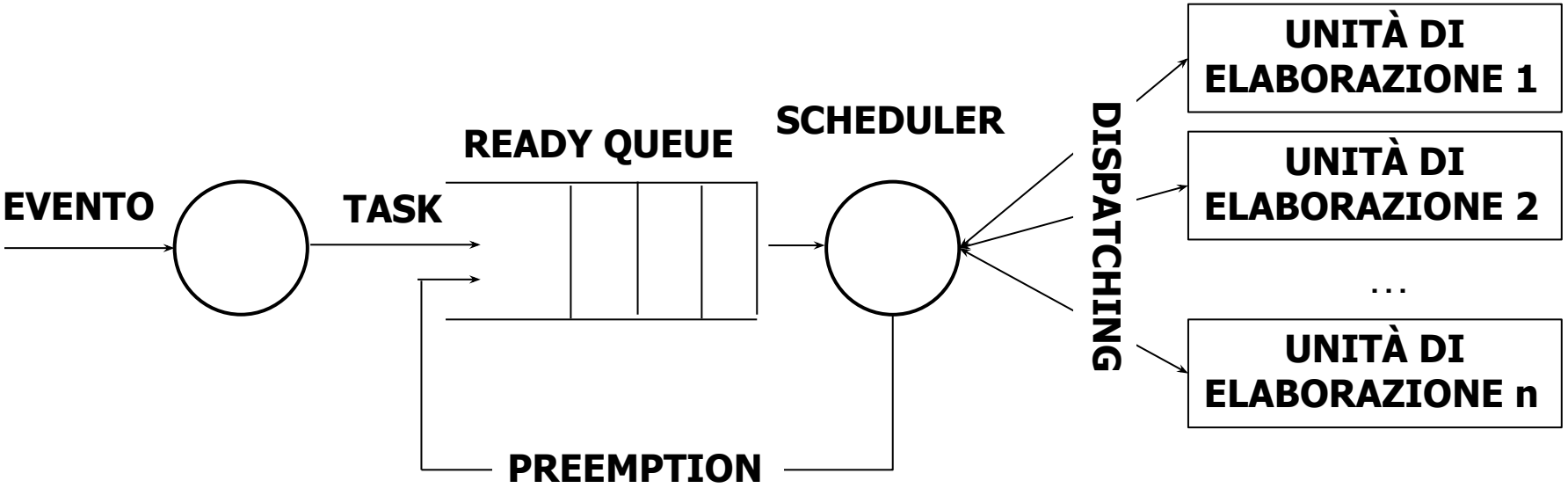
Lo scheduling interviene per associare un task presente nella coda ad una delle unità di elaborazione disponibili (**DISPATCHING**).



RUOLO DELLO SCHEDULING

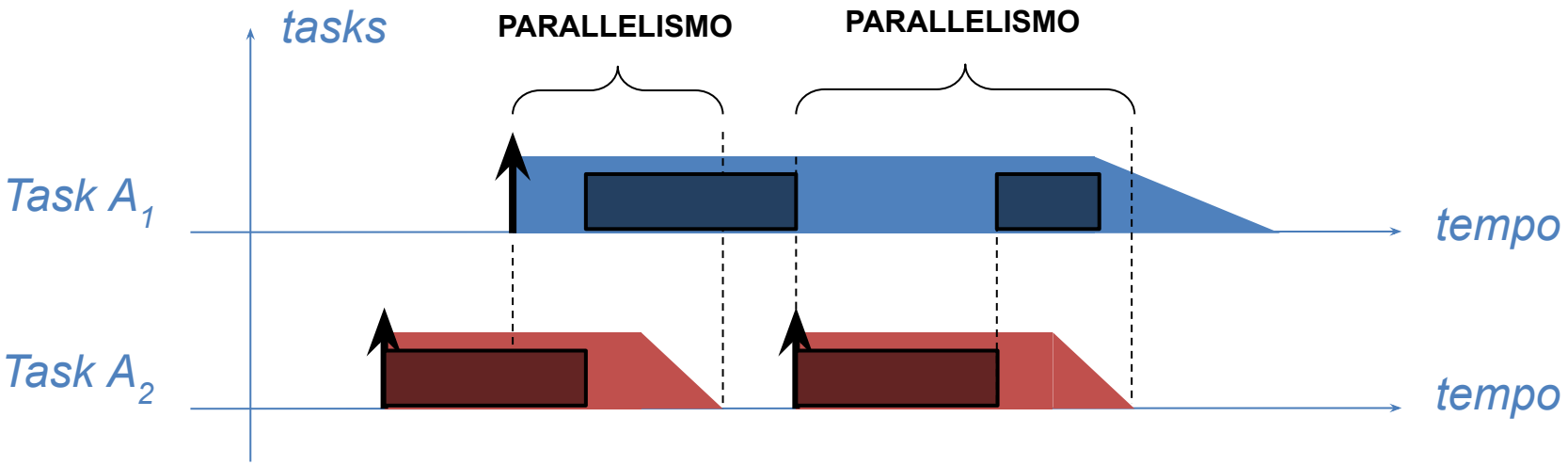
In situazioni di parallelismo può capire che task a **maggiore priorità** sottraggono le risorse di elaborazione associate a task con **minore priorità**.

Tale meccanismo è chiamato **PREEMPTION** e rispetta il **VINCOLO DI PRECENDEZA** tra task.

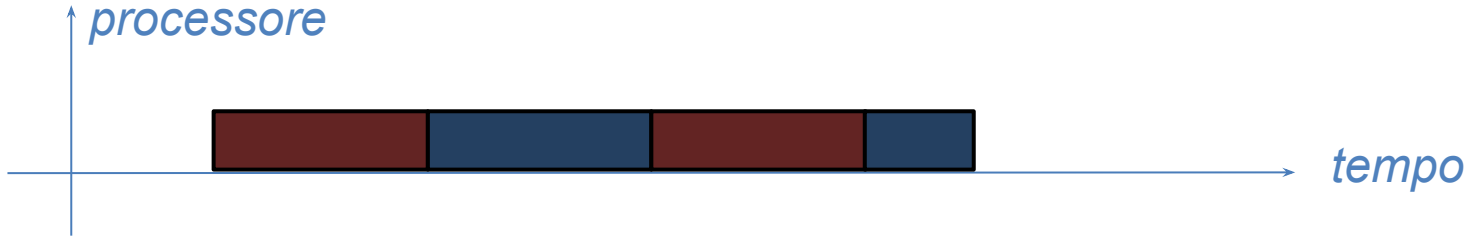


ESEMPIO DI SCHEDULING

Ipotizzando di avere un sistema MONOPROCESSORE e A_1 con minor priorità di A_2:

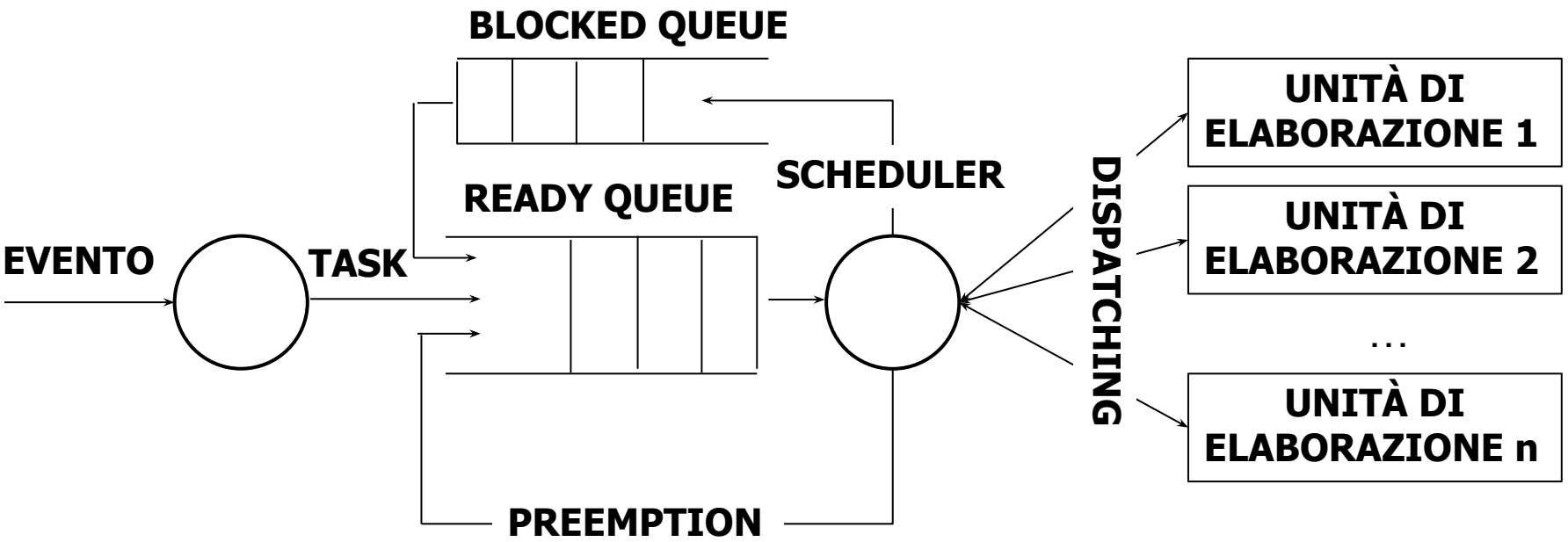


Una possibile soluzione del problema di scheduling può essere:



RUOLO DELLO SCHEDULING

In situazioni di parallelismo può capitare che due o più task attivi siano costretti ad operare uno alla volta perché condividono una stessa risorsa che può essere utilizzata solo da un task alla volta. In questo caso lo scheduler deve rispettare un **VINCOLO DI MUTUA ESCLUSIONE SU RISORSE CONDIVISE** e mettere i task attivi concorrenti in una coda a parte (**BLOCKED QUEUE**)



VINCOLO DI PRECEDENZA vs MUTUA ESCLUSIONE

Ipotizziamo di avere un sistema MONOPROCESSORE deputato al coordinamento di elementi singoli (differenti task) che, per questioni di gestione dell'impianto, eseguono un log costante ad un'unica stampante condivisa.

In questo caso l'applicazione di uno scheduling con preemption non è sufficiente a garantire una **correttezza logica** del funzionamento dell'impianto.

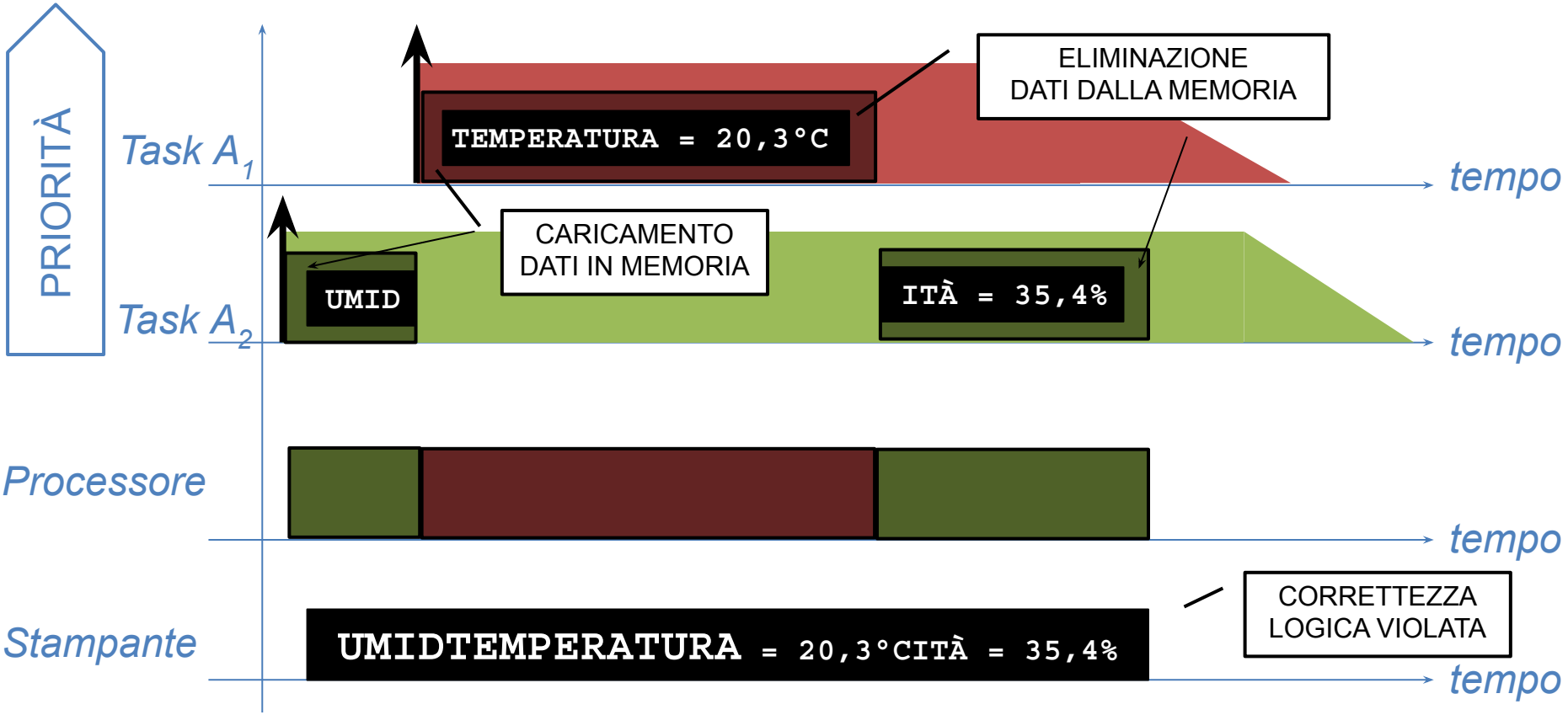
Si ipotizzino due Task, A_1 e A_2 il primo dei quali deve stampare la stringa «Temperatura = 20,3°C» ed il secondo deve stampare la stringa «Umidità = 35,4%»

Supponiamo inoltre che:

- il task A_1 abbia maggior priorità rispetto al task A_2;
- il task A_1 diventi attivo dopo il task A_2;

VINCOLO DI PRECEDENZA vs MUTUA ESCLUSIONE

Dal punto di vista grafico avremo:



VINCOLO DI PRECENDENZA E DI MUTUA ESCLUSIONE

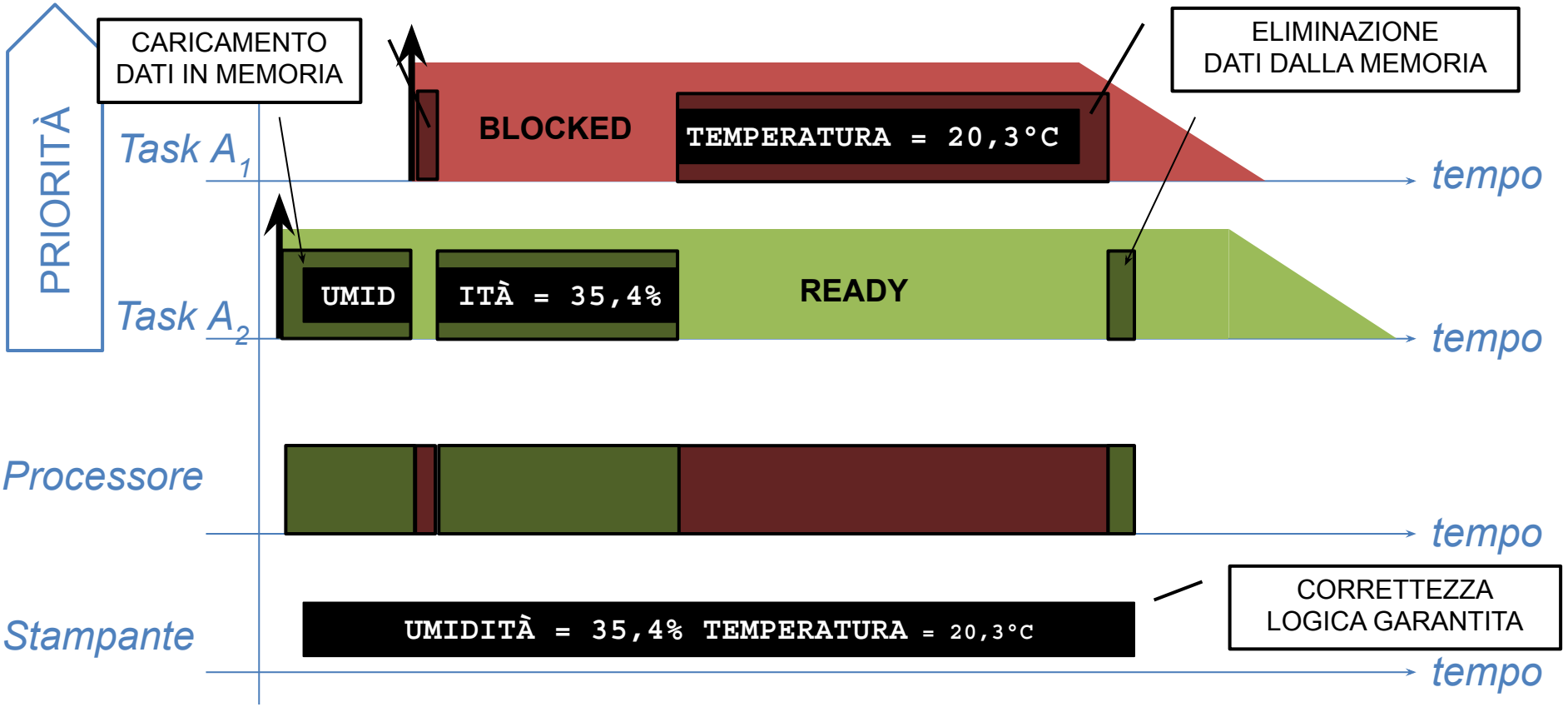
Per risolvere correttamente il problema di scheduling è necessario mantenere sia la coda dei task attivi (READY QUEUE) che la cosa dei task bloccati (BLOCKED QUEUE).

Il tentativo di accesso alla risorsa condivisa già in uso da un altro processo deve essere bloccato fino alla liberazione della risorsa stessa.

Nel nostro esempio la risorsa condivisa è la STAMPANTE e per garantire il vincolo di correttezza logica è necessario applicare un algoritmo di scheduling con preemption e con gestione della blocked queue.

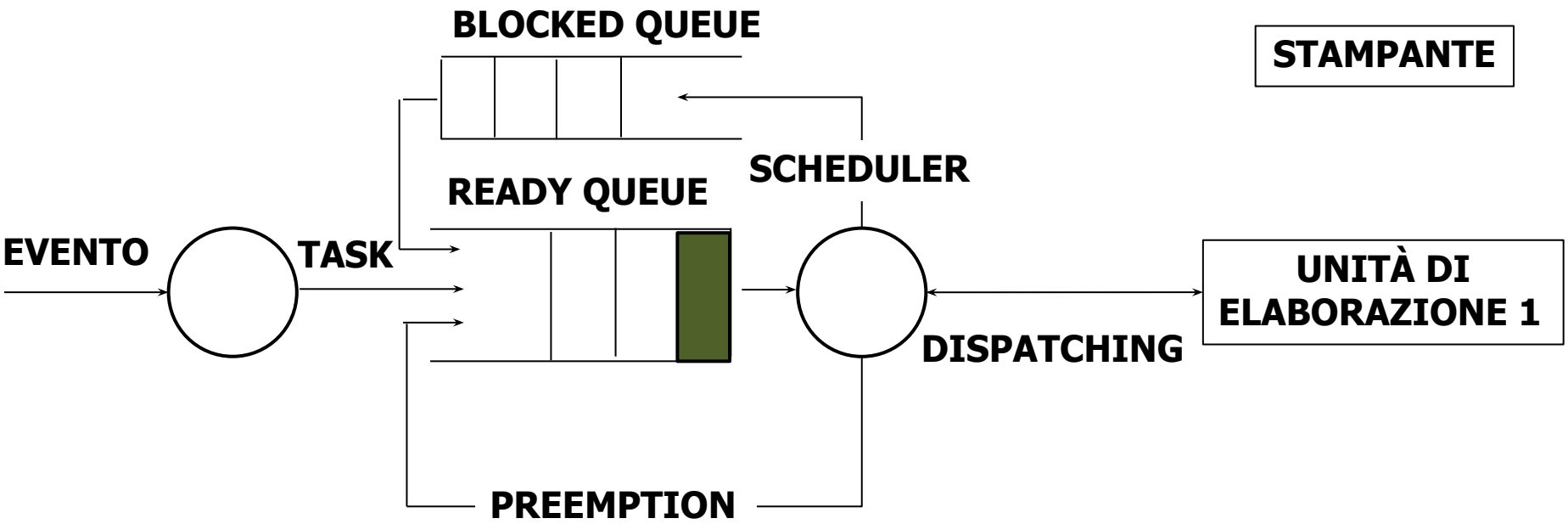
VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

Dal punto di vista grafico avremo:



VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

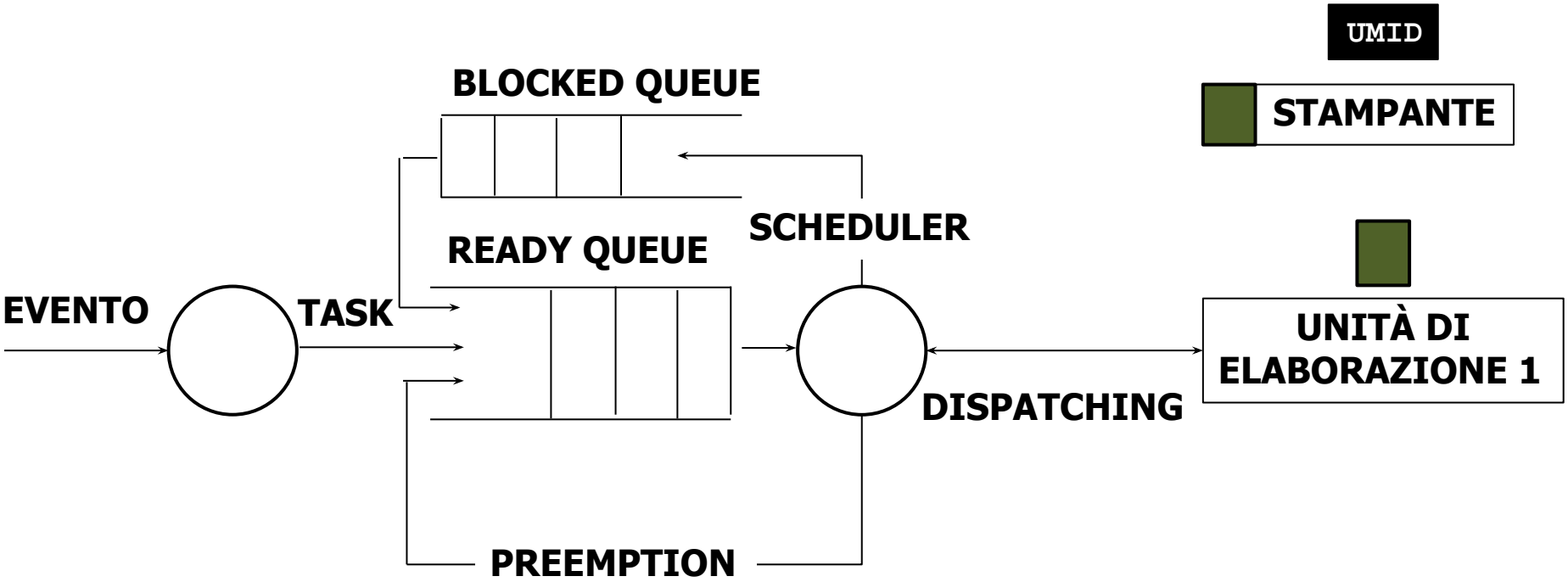
Il primo evento è quello che attiva il Task A_2 che viene messo nella «READY QUEUE»



VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

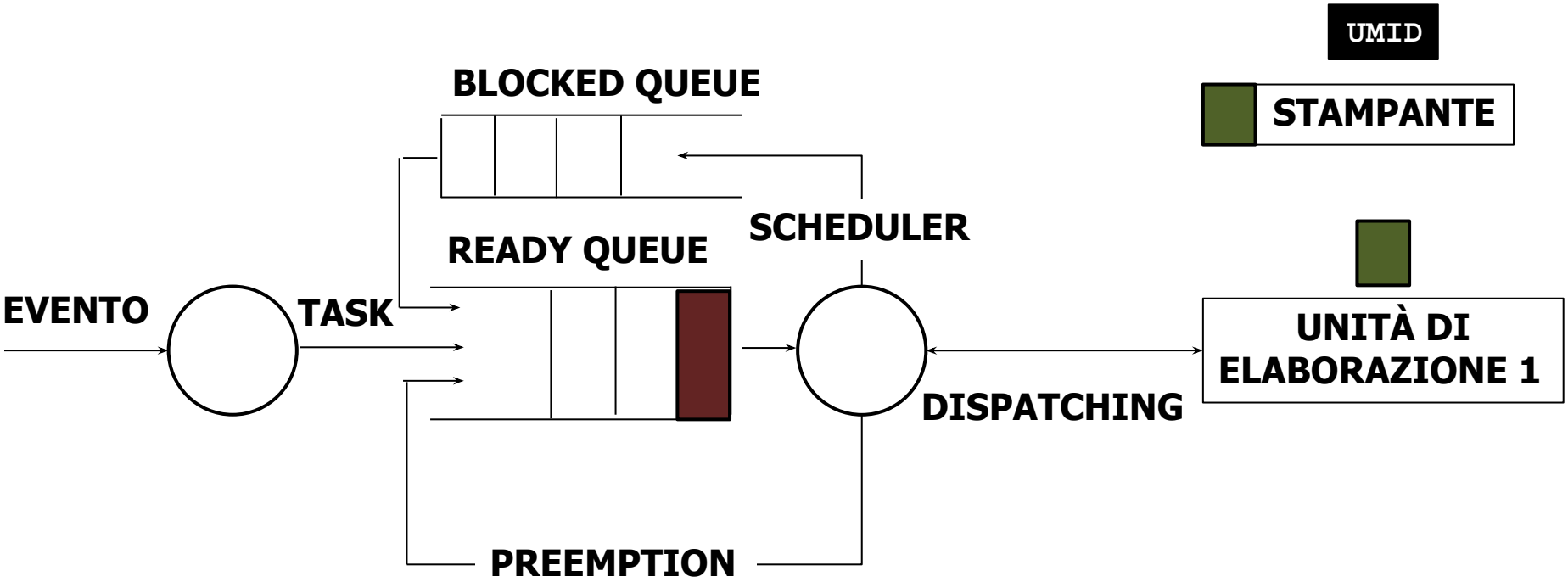
Lo scheduler manda in esecuzione il Task A_2 visto che la risorsa di elaborazione è disponibile.

Il Task A_2 carica i dati nella memoria della stampante e avvia la stampa dei caratteri della stringa.



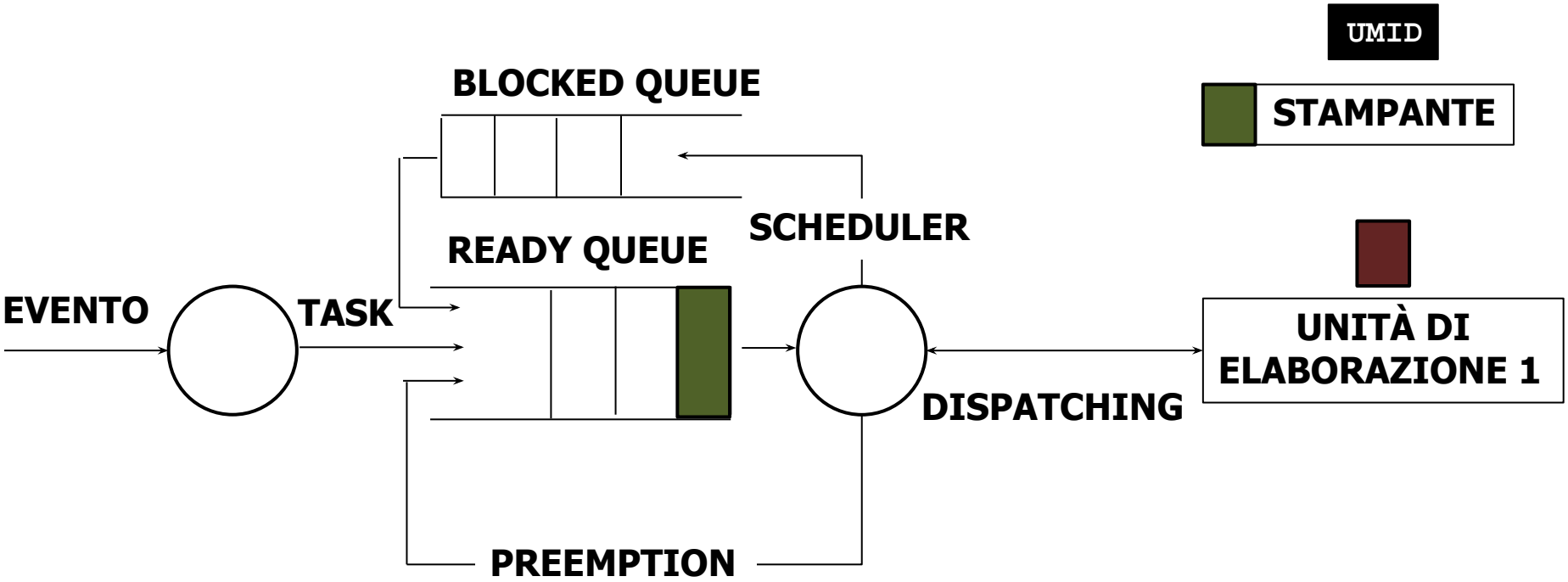
VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

Mentre la stampante sta operando, un nuovo evento attiva il Task A_1 che viene collocato nella READY QUEUE.



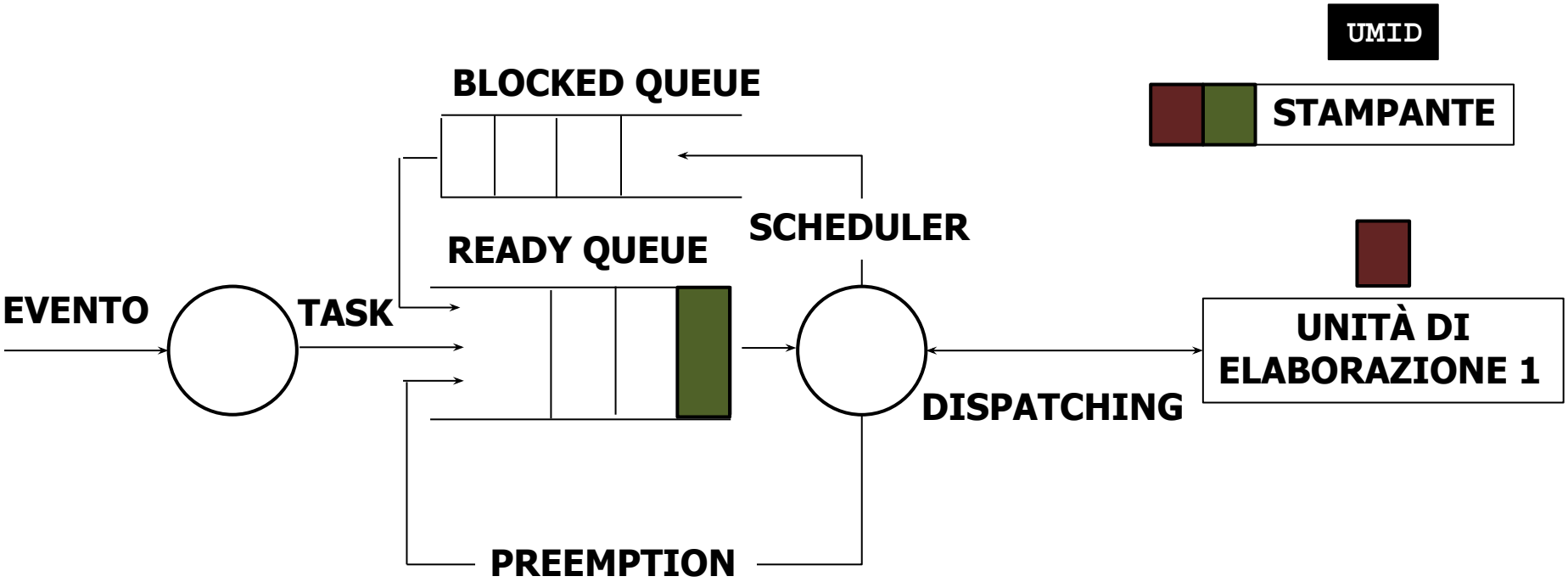
VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

Il Task A_1 ha maggiore priorità rispetto al Task A_2 pertanto lo scheduler fa PREEMPTION e manda all'unità di elaborazione il Task A_1 e sposta il Task A_2 nella READY QUEUE.



VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

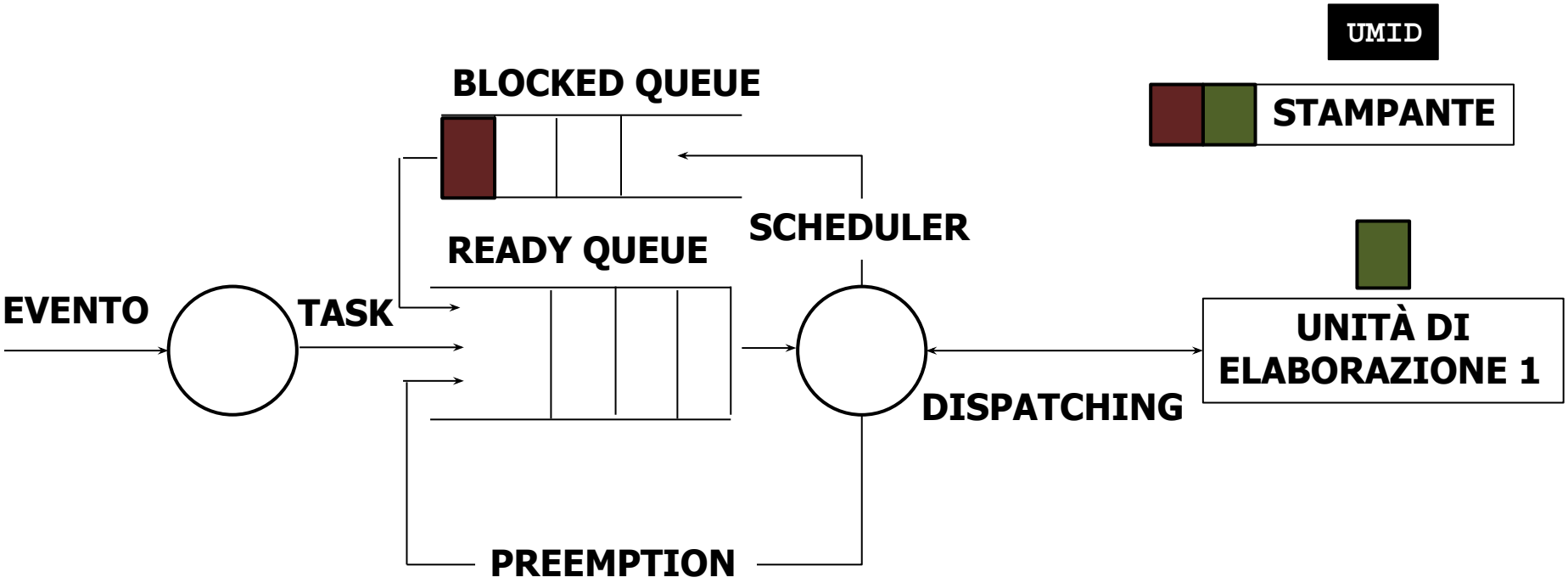
Il Task A_1 carica i dati di stampa nella stampante.



VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

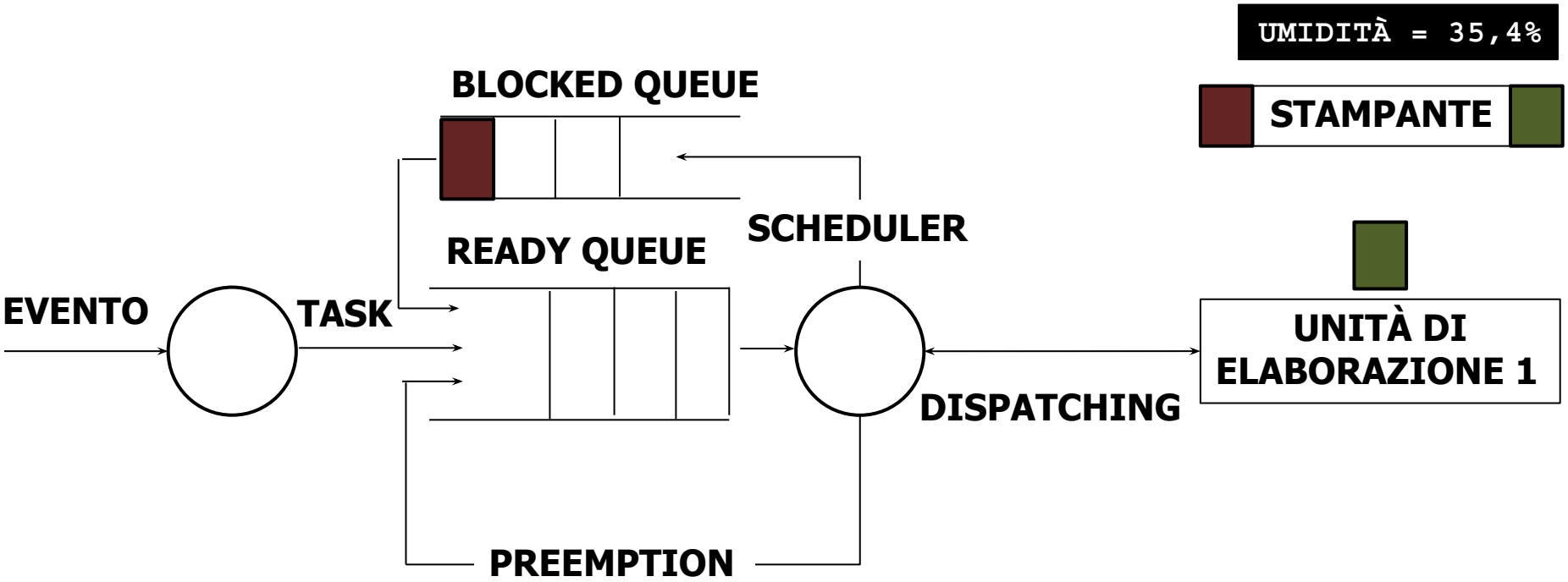
Quando il Task A_1 tenta di stampare, la risorsa è già occupata dal Task A_2 pertanto lo scheduler applica il vincolo di mutua esclusione agendo in questa sequenza:

- Sposta il Task A_1 nella BLOCKED QUEUE;
- Sposta il Task A_2 nella unità di elaborazione;



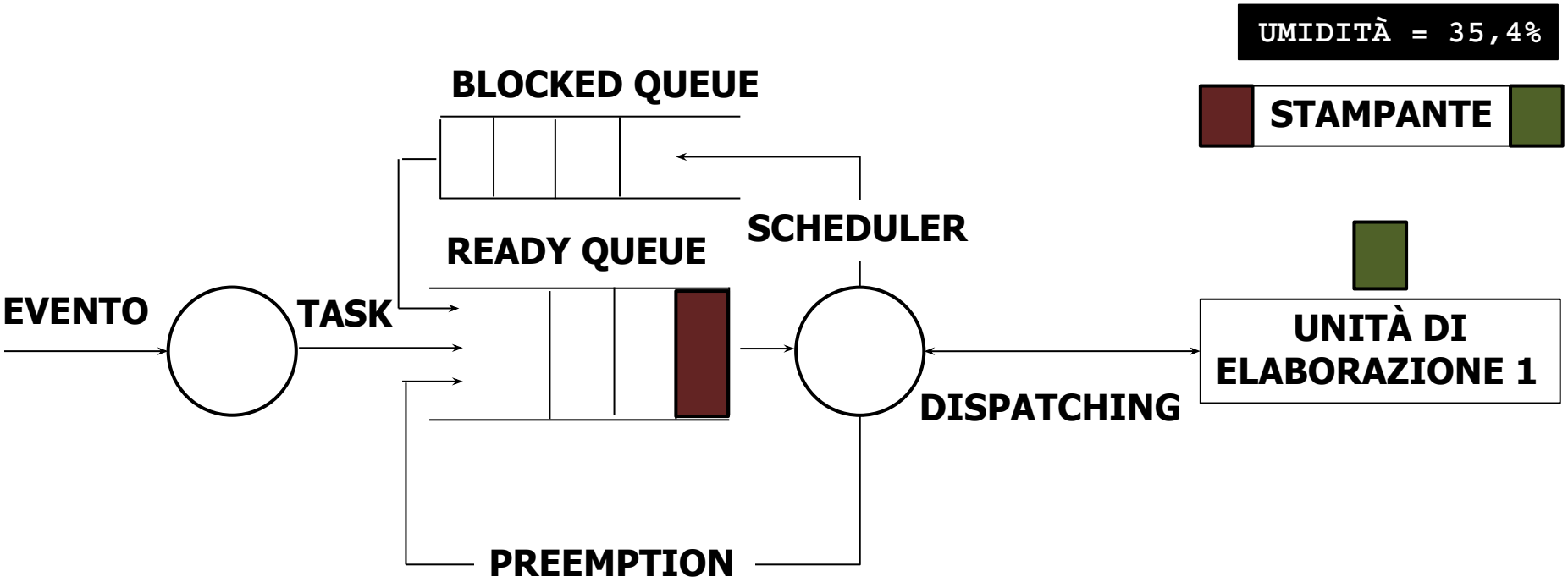
VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

Quando il Task A_2 completa la stampa, libera la risorsa stampante.



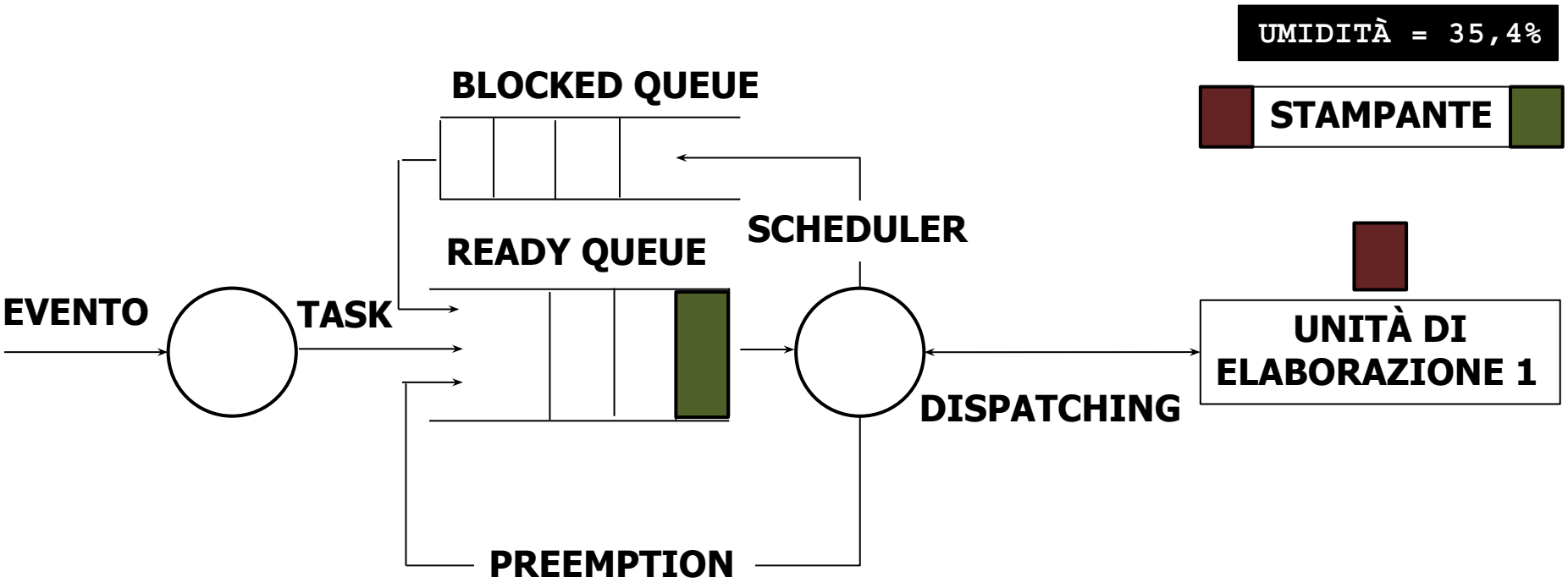
VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

Lo scheduler sposta il Task A_1 dalla BLOCKED QUEUE alla READY QUEUE



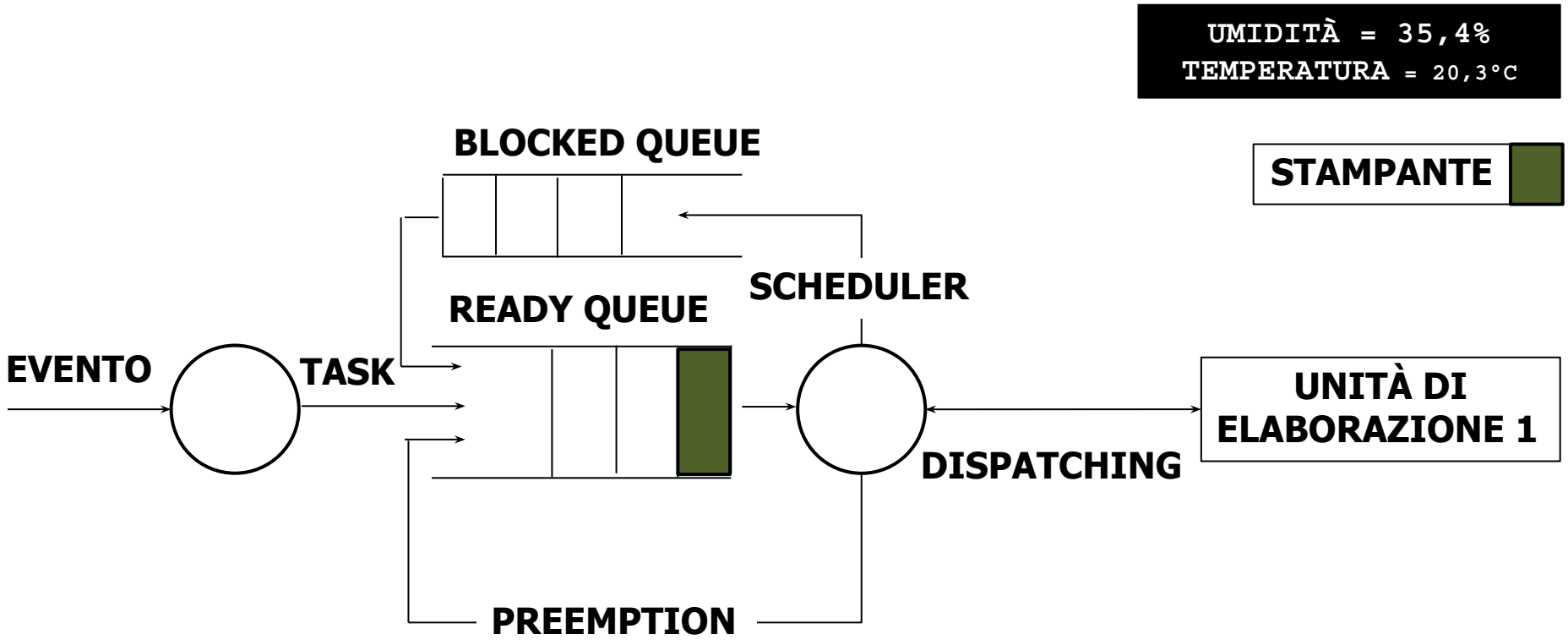
VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

Il Task A_1 ha maggiore priorità rispetto al Task A_2 pertanto lo scheduler fa PREEMPTION e manda all'unità di elaborazione il Task A_1 e sposta il Task A_2 nella READY QUEUE.



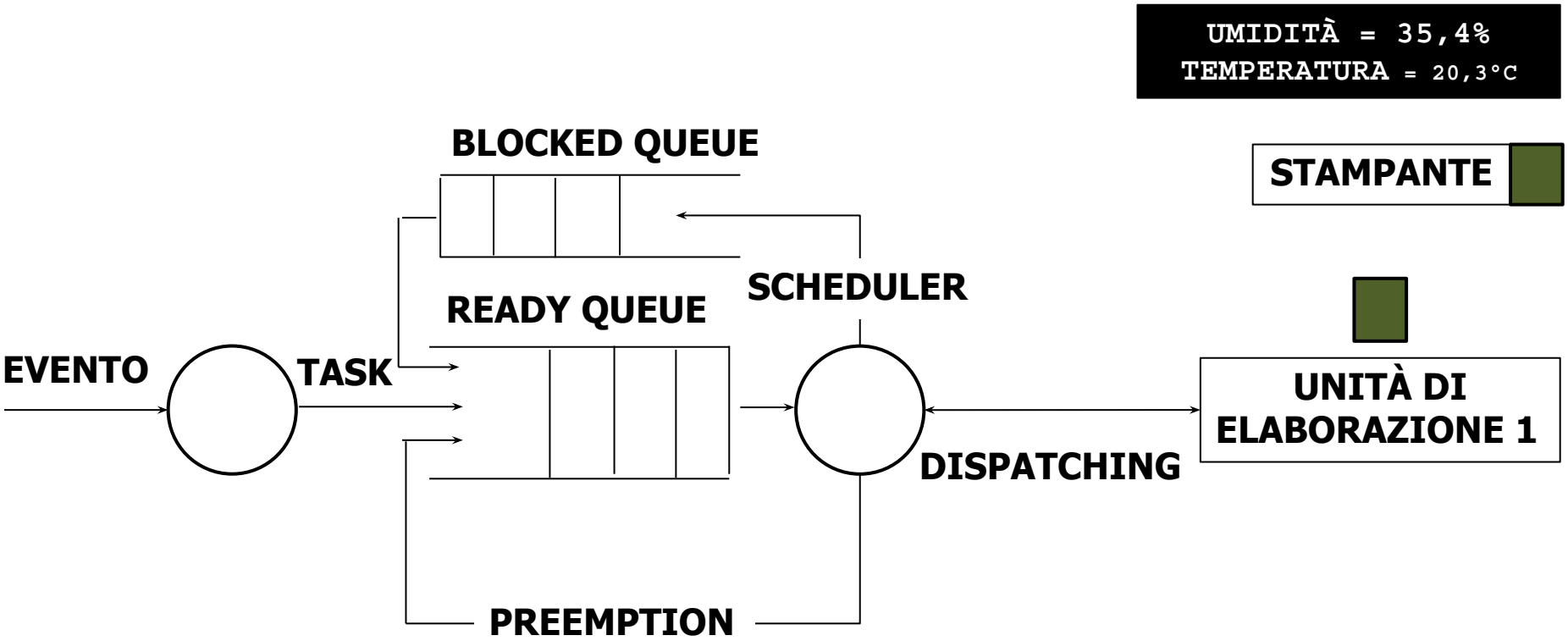
VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

Il Task A_1 può finalmente stampare e terminare completamente tutte le attività, liberando tutte le risorse (stampante ed unità di elaborazione).



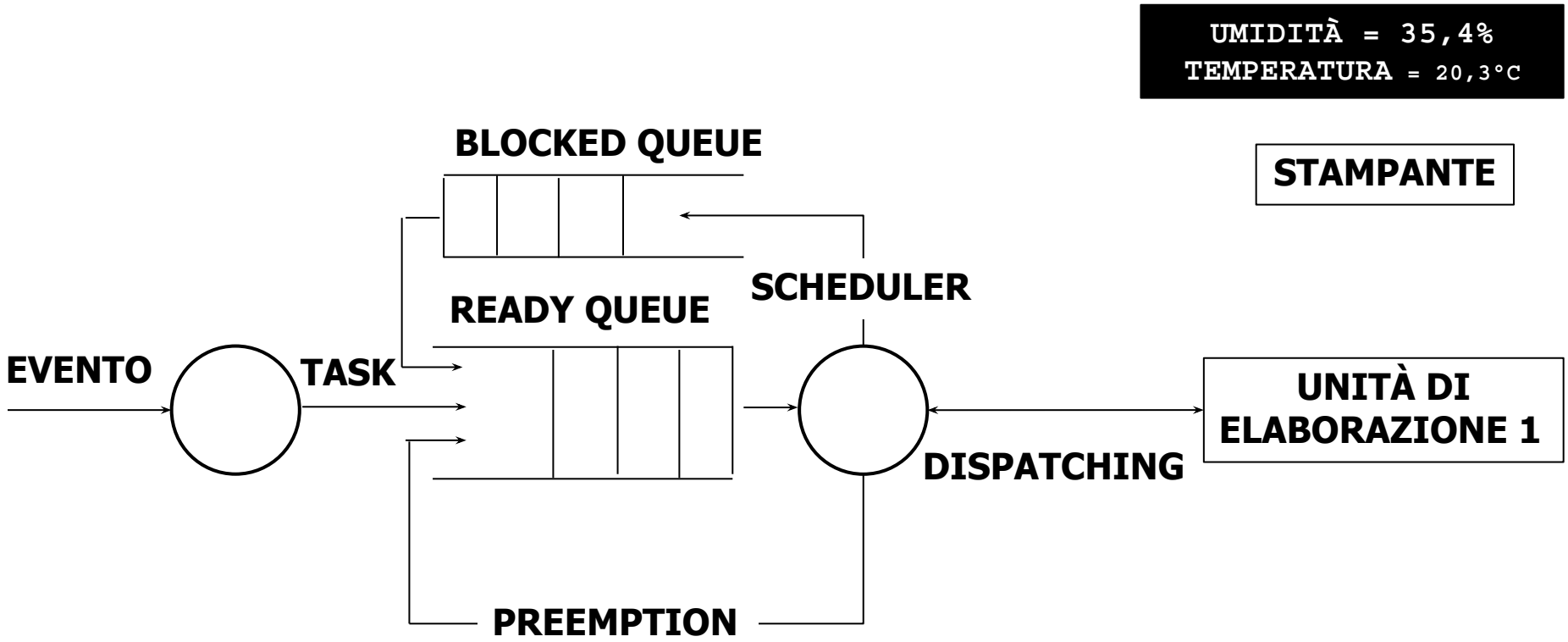
VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

Lo scheduler sposta il Task A_2 nella unità di elaborazione.



VINCOLO DI PRECEDENZA E DI MUTUA ESCLUSIONE

Il Task A_2 può finalmente terminare completamente tutte le attività, liberando tutte le risorse (stampante ed unità di elaborazione).



BIBLIOGRAFIA

Sezione 2.1, 2.2 e 2.3

TITOLO

Sistemi di automazione industriale

Architetture e controllo

AUTORI

Claudio Bonivento

Luca Gentili

Andrea Paoli

EDITORE

McGraw-Hill