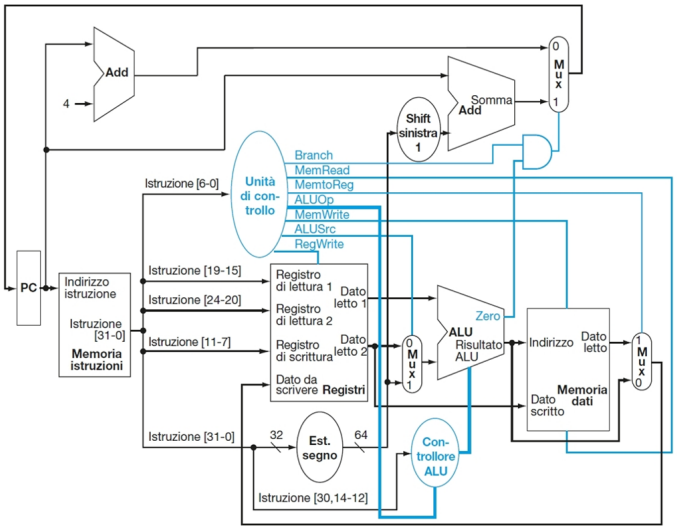

Figure 1: Schema di datapath

Rispondere alle domande a risposta multipla annerendo la casella corrispondente alla risposta corretta. Ogni domanda ha una ed una sola risposta corretta.

Cognome e Nome: `Nome10 Cognome11`

Numero di Matricola: `00`

Domanda 1 Inizialmente il contenuto di `x5` = 0x0000000000000000 Quale valore conterrá `x5` dopo aver eseguito le seguenti istruzioni Assembly RISC-V?

```
addi x6, x0, 0x000F
slli x6, x6, 28
or x5, x5, x6
```

- ☐ `x5` = 1111 0000 0000 0000 0000 0000 0000 0000 1111 0000 0000 0000 0000 0000 0000 1111_2
- ☐ `x5` = 1111 0000 0000 0000 0000 0000 0000 0000 1111 1111 0000 0000 0000 0000 0000 0000_2
- ☐ Nessuna delle altre risposte
- ■ `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 1111 0000 0000 0000 0000 0000 0000 0000_2
- ☐ `x5` = 0000 1111 0000 0000 0000 0000 0000 1111 1111 0000 0000 0000 0000 0000 0000 0000_2

Domanda 2 Supponiamo che i vari blocchi logici richiesti per implementare l'unità di elaborazione del processore RISC-V (si veda Figura 1) abbiano le seguenti latenze:

- Mem-I/Mem-D 250ps,
- Register File 150ps,
- Mux 25ps,
- ALU 200ps,
- Addizionatore 150ps,
- Porta Logica singola (e.g. AND) 15ps,
- Lettura Registro 40ps, (Con Lettura Registro si intende il tempo che intercorre tra il fronte di salita del clock e l'istante in cui il valore contenuto nel registro compare sull'uscita del tegistro stesso. Questo tempo si applica solo al PC.)
- Impostazione Registro 14ps, (Impostazione Regsitro si riferisce al tempo per cui il dato in input ad un registro deve rimanere stabile prima del fronte di salita del clock. Questo tempo si applica sia al PC che al Register File).

- Estensione segno 45ps,
- Controllo 45ps.

Quale è la latenza di una istruzione di tipo R (Cioè quanto deve durare il periodo del clock per essere sicuri che l'istruzione venga eseguita correttamente. Attenzione a considerare solo i multiplexer sul cammino critico.)?

- ☐ 40 + 150 + 25 + 200 + 25 + 14 = 454ps
- ■ 40 + 250 + 150 + 25 + 200 + 25 + 14 = 704ps
- ☐ 40*3 + 250 + 150 + 25 + 200 + 25 + 14*3 = 812ps
- ☐ Nessuna delle altre risposte.
- ☐ 40 + 250 + 150 + 25 + 200 + 25 + 14*3 = 732ps

Domanda 3

```
int mathforfun(int i, int j,
               int q, int s) {
  for(i=0;i<q; i++)
  {
     q=q+s;
     q=q-j;
  }
  return q;
}
```

La funzione mathforfun prende in ingresso quattro argomenti e al suo interno svolge con questi delle operazioni matematiche ritornando un intero. Data la traduzione parziale in assembly intel qui sotto come completereste le righe X1 e X2 mancanti? scegliere una delle opzioni

```
mathforfun:
        pushq   %rbp
        movq    %rsp, %rbp
        movl    %edi, -4(%rbp)
        movl    %esi, -8(%rbp)
        movl    %edx, -12(%rbp)
        movl    %ecx, -16(%rbp)
        movl    $0, -4(%rbp)
.L3:
        movl    -4(%rbp), %eax
        X1
        jge     .L2
        movl    -16(%rbp), %eax
        addl    %eax, -12(%rbp)
        movl    -8(%rbp), %eax
        subl    %eax, -12(%rbp)
        X2
        jmp     .L3
.L2:
        movl    -12(%rbp), %eax
        popq    %rbp
        ret
```

- ☐ X1: `addl` $1, -8(%rbp)
  X2: `cmpl` -12(%rbp), %eax
- ☐ X1: `movl` $1, -8(%rbp)
  X2: `cmpl` -10(%rbp), %eax
- ☐ Nessuna delle altre risposte
- ☐ X1: `addl` $1, -4(%rbp)
  X2: `cmpl` -12(%rcx), %eax
- ■ X1: `cmpl` -12(%rbp), %eax
  X2: `addl` $1, -4(%rbp)

Domanda 4 Quale delle seguenti affermazioni è FALSA?

- [ ] Il linguaggio Assembly è strettamente legato alla CPU su cui il programma dovrà eseguire.
- [ ] Per essere eseguito, un programma Assembly deve essere tradotto in linguaggio macchina da un compilatore.
- [x] Una caratteristica dei programma scritti in linguaggio Assembly è la sua portabilità.
- [ ] Il linguaggio Assembly codifica le istruzioni macchina tramite codici mnemonici.
- [ ] Al livelo più basso, la CPU può capire solo programmi scritti in linguaggio macchina.

Domanda 5 Si dica a quale delle seguenti alternative corrisponde la seguente istruzione in assembly ARM:

```
mov  r7, r5, lsl #2
```

- [x] `r7 = 4 * r5;`
- [ ] `r7 = 2 * r5;`
- [ ] Nessuna delle altre risposte
- [ ] `r5 = 2 * r7;`
- [ ] `r5 = 4 * r7;`

Domanda 6 Usando la rappresentazione binaria, svolgere la sottrazione $10110011101100011001 - 0x4AA95$

- [x] 01101001000010000100_2
- [ ] 10111001100000001000_2
- [ ] 10110001000100001000_2
- [ ] Nessuna delle altre risposte
- [ ] 01100001000010010100_2

Domanda 7 Si consideri una CPU che impiega $100ps$ per la fase di fetch, $150ps$ per la fase di decodifica, $200ps$ per eseguire operazioni con la ALU, $150ps$ per la fase di accesso alla memoria e $100ps$ per la fase di scrittura nel register file. L'incremento di prestazioni che ci si può attendere usando una pipeline è:

- [x] Nessuna delle altre risposte
- [ ] di 2 volte
- [ ] di 2.5 volte
- [ ] di 3 volte
- [ ] di 4 volte

Domanda 8 Si consideri una CPU dotata di 2 cache separate per dati ed istruzioni. Il CPI ideale della CPU è 4, la cache istruzioni ha una frequenza di miss del 1% e la cache dati ha una frequenza di miss del 4%. Supponendo che un cache miss richieda 100 cicli di clock per essere servito e che il 20% delle istruzioni Assembly accedano a dati in memoria, il CPI reale (il numero di cicli necessari in media per eseguire un'istruzione tenendo conto degli stalli per accesso alla RAM) è:

- [x] Nessuna delle altre risposte
- [ ] 8
- [ ] 3.72
- [ ] 3.44
- [ ] 7.44

Domanda 9 Svolgere la somma dei due numeri espressi in complemento a 2 su 12 bit $0111\,1011\,0101_2 + 1100\,0100\,1010_2$

- [ ] 5119_{10}
- [ ] -5119_{10}
- [x] 1023_{10}
- [ ] Nessuna delle altre risposte

☐ -1023_{10}

Domanda 10 Considerare l'operazione di somma tra interi in assembly RISC-V. Essa puó essere richiesta:

☐ Con due operandi sorgenti registri e un operando destinazione in memoria

☐ Nessuna delle altre risposte

■ Con due operandi sorgenti di tipo registro e un operando destinazione di tipo registro, oppure con un operando di tipo registro e un operando immediato come sorgenti e un operando registro destinazione

☐ Con due operandi sorgenti immediati e un operando destinazione registro

☐ Con due operandi sorgenti registri e una destinazione anche esso registro

Domanda 11 Convertire in decimale il binario a virgola fissa 1001110.0011_2

☐ Nessuna delle altre risposte

☐ $1001110.0011_2 = 156.1875_{10}$

☐ $1001110.0011_2 = 78.375_{10}$

■ $1001110.0011_2 = 78.1875_{10}$

☐ $1001110.0011_2 = 156.375_{10}$

Domanda 12

Si consideri la seguente funzione nel linguaggio C chiamata "sort" il cui scopo e' quello di ordinare un array in ingresso. Tale funzione prende in ingresso un array v[] (espresso naturalmente come puntatore a long long int) e la lunghezza n del vettore.
Al suo interno la funzione esegue una chiamata ad un'altra funzione denominata "swap" che scambia il valore dell'elemento del vettore in ingresso in posizione k con l'elemento successivo k+1. Quale delle implementazioni in assembly RISC-V della funzione swap e' corretta tra quelle proposte?

```
void sort (long long int v[], long long int n){
  long long int i, j;
  for (i=0; i<n; i+=1) {
   for (j=i-1; j>=0 && v[j] > v[j+1]; j-=1) {
    swap (v,j);
   }
  }
}

void swap (long long int v[], long long int k){
  long long int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
 }
 }
```

```
sort:  ...
       ...
       ...
       ...
       jal  swap
       ...
       ...
       ...

swap:  ...
       ...
       ...
       ...
       ...
       ...
       ...
```

☐ Nessuna delle altre risposte

☐
```
swap:  slli  x7, x11, 3
       ld    x5, 0(x10)
       ld    x6, 8(x10)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       add   x7, x10, x7
```

☐
```
swap:  srli  x7, x11, 3
       add   x7, x10, x7
       ld    x5, 0(x7)
       ld    x6, 8(x7)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       jalr  x0, 0(x1)
```

☐
```
swap:  slli  x7, x11, 4
       add   x7, x10, x7
       ld    x7, 8(x7)
       sd    x5, 0(x7)
       jalr  x0, 0(x1)
```

■
```
swap:  slli  x7, x11, 3
       add   x7, x10, x7
       ld    x5, 0(x7)
       ld    x6, 8(x7)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       jalr  x0, 0(x1)
```

Rispondere alle domande a risposta multipla annerendo la casella corrispondente alla risposta corretta. Ogni domanda ha una ed una sola risposta corretta.

Cognome e Nome: Nome21 Cognome21

Numero di Matricola: 11

Domanda 1

```
int mathforfun(int i, int j,
               int q, int s) {
  for(i=0;i<q; i++)
  {
     q=q+s;
     q=q-j;
  }
  return q;
}
```

La funzione mathforfun prende in ingresso quattro argomenti e al suo interno svolge con questi delle operazioni matematiche ritornando un intero. Data la traduzione parziale in assembly intel qui sotto come completereste le righe X1 e X2 mancanti? scegliere una delle opzioni

```
mathforfun:
            pushq   %rbp
            movq    %rsp, %rbp
            movl    %edi, -4(%rbp)
            movl    %esi, -8(%rbp)
            movl    %edx, -12(%rbp)
            movl    %ecx, -16(%rbp)
            movl    $0, -4(%rbp)
.L3:
            movl    -4(%rbp), %eax
            X1
            jge     .L2
            movl    -16(%rbp), %eax
            addl    %eax, -12(%rbp)
            movl    -8(%rbp), %eax
            subl    %eax, -12(%rbp)
            X2
            jmp     .L3
.L2:
            movl    -12(%rbp), %eax
            popq    %rbp
            ret
```

- ☐ Nessuna delle altre risposte
- ■ X1: `cmpl` -12(%rbp), %eax
  X2: `addl` $1, -4(%rbp)
- ☐ X1: `addl` $1, -4(%rbp)
  X2: `cmpl` -12(%rcx), %eax
- ☐ X1: `movl` $1, -8(%rbp)
  X2: `cmpl` -10(%rbp), %eax
- ☐ X1: `addl` $1, -8(%rbp)
  X2: `cmpl` -12(%rbp), %eax

Domanda 2 Supponiamo che i vari blocchi logici richiesti per implementare l'unità di elaborazione del processore RISC-V (si veda Figura 1) abbiano le seguenti latenze:

- Mem-I/Mem-D 250ps,
- Register File 150ps,
- Mux 25ps,

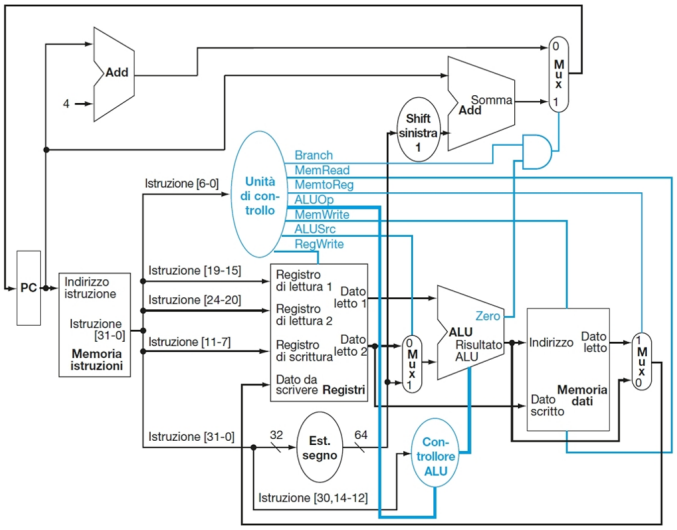

Figure 1: Schema di datapath

- ALU 200ps,
- Addizionatore 150ps,
- Porta Logica singola (e.g. AND) 15ps,
- Lettura Registro 40ps, (Con Lettura Registro si intende il tempo che intercorre tra il fronte di salita del clock e l'istante in cui il valore contenuto nel registro compare sull'uscita del tegistro stesso. Questo tempo si applica solo al PC.)
- Impostazione Registro 14ps, (Impostazione Regsitro si riferisce al tempo per cui il dato in input ad un registro deve rimanere stabile prima del fronte di salita del clock. Questo tempo si applica sia al PC che al Register File).
- Estensione segno 45ps,
- Controllo 45ps.

Quale è la latenza di una istruzione di tipo R (Cioè quanto deve durare il periodo del clock per essere sicuri che l'istruzione venga eseguita correttamente. Attenzione a considerare solo i multiplexer sul cammino critico.)?

- ☐ 40 + 150 + 25 + 200 + 25 + 14 = 454ps
- ☐ Nessuna delle altre risposte.
- ☐ 40*3 + 250 + 150 + 25 + 200 + 25 + 14*3 = 812ps
- ■ 40 + 250 + 150 + 25 + 200 + 25 + 14 = 704ps
- ☐ 40 + 250 + 150 + 25 + 200 + 25 + 14*3 = 732ps

Domanda 3 Quali delle seguenti singole istruzioni assembly RISC-V equivale alle due istruzioni

```
add x6, x5, x10
ld x7, 0(x6)
```

- ☐ `mv x6, x7`
- ☐ Tutte le risposte si equivalgono
- ☐ `add x6, x7, x5`
- ■ Nessuna delle altre risposte
- ☐ `ld x6, x7(x5)`

Domanda 4 Si consideri una CPU che impiega $600ps$ per la fase di fetch, $600ps$ per la fase di decodifica, $500ps$ per eseguire operazioni con la ALU, $400ps$ per la fase di accesso alla memoria e $700ps$ per la fase di scrittura nel register file. Il massimo incremento di prestazioni che ci si può attendere usando una pipeline è:

- [ ] di 2 volte
- [ ] di 3 volte
- [ ] Nessuna delle altre risposte
- [ ] di 2.5 volte
- [x] di 4 volte

Domanda 5 Il registro `x5` contiene il valore `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1100 1100 1111 1111 0011 0000. Quale valore conterrá `x5` dopo aver eseguito le seguenti istruzioni Assembly RISC-V?

```
srli x6, x5, 4
slli x5, x5, 10
and x5, x5, x6
```

- [ ] `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 0011 0011 1111 1100 1100 0000 0000 0000_2
- [x] `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1100 1100 0000 0000 0000_2
- [ ] `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 0001 1100_2
- [ ] `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1100 1100 1111 1111 0011_2
- [ ] Nessuna delle altre risposte

Domanda 6 Usando la rappresentazione binaria, svolgere la sottrazione $10110011101100011001 - 0x4AA95$

- [ ] 01100001000010010100_2
- [ ] Nessuna delle altre risposte
- [ ] 10110001000100001000_2
- [x] 01101001000010000100_2
- [ ] 10111001100000001000_2

Domanda 7 Svolgere in complemento a 2 su 4 bit l'operazione $1110_2 - 5_{10}$

- [ ] 1111_2
- [ ] 0111_2
- [x] 1001_2
- [ ] 1000_2
- [ ] Nessuna delle altre in quanto il risultato non e' rappresentabile in CA2 con soli 4 bit

Domanda 8 Come si rappresenta in decimale il numero binario 00100110_2?

- [ ] Nessuna delle altre risposte
- [ ] 100_{10}
- [x] 38_{10}
- [ ] 76_{10}
- [ ] 25_{10}

Domanda 9 La funzione `sumV` prende in ingresso l'indirizzo di tre array di interi `u`, `v`, `z`, e la dimensione `size` degli array, il suo compito è quello di sommare gli elementi nella posizione i-esima di `u` e `v` e di salvare il risultato della somma nella posizione i-esima di `z`. Il compilatore fornisce la seguente traduzione in assembly ARM che è incompleta: le righe X1, X2 e X3 sono omesse. (Sono rispettate le convenzioni di chiamata per gli argomenti e per il valore di ritorno specificate dall'ABI vista durante il corso) Quale delle coppie X1, X2 e X3 proposte corrisponde alle righe corrette?

```
void sumV(int * u, int * v, int* z, unsigned int size){
  for (int i = size - 1; i >= 0; i--) {
    *(z+i) = *(v+i) + *(u+i);
  }
  return;
}
```

```
sumV:
        stmfd   sp!, {r11, lr}
        mov     r11, sp
        X1
.LBB01:
        X2
        sub     lr, lr, #1
        X3
```

X1
```
subs lr, r3, #1
ldmfdmi sp!, {r11, lr}
```
X2
```
ldr r12, [r1], lr, lsl #2
ldr r3, [r0], lr, lsl #2
add r3, r3, r12
str r3, [r2], lr, lsl #2
```
X3
```
cmn lr, #1
bgt .LBB01
ldmfd sp!, {r11, lr}
```

X1
```
subs lr, r3, #1
ldmfdmi sp!, {r11, pc}
```
X2
```
ldr r12, [r1, lr, lsl #2 ]
ldr r3, [r0, lr, lsl #2 ]
add r3, r3, r12
str r3, [r2, lr, lsl #2 ]
```
X3
```
cmn lr, #1
bgt .LBB01
ldmfd sp!, {r11, pc}
```

X1
```
subs lr, r3, #1
ldmfdmi sp!, {r11, pc}
```
X2
```
ldr r12, [r1, lr, lrl #2 ]
ldr r3, [r0, lr, lrl #2 ]
add r3, r3, r12
str r3, [r2, lr, lrl #2 ]
```
X3
```
cmn lr, #1
bgt .LBB01
ldmfd sp!, {r11, pc}
```

☐ Nessuna delle altre risposte

X1
```
sub lr, r3, #1
ldmfdmi sp!, {r11, pc}
```
X2
```
ldr r12, [r1, lr, lsl #2 ]
ldr r3, [r0, lr, lsl #2 ]
add r3, r3, r12
str r3, [r2, lr, lsl #2 ]
```
X3
```
cmp lr, #1
bge .LBB01
ldmfd sp!, {r11, pc}
```

Domanda 10 Il principio di *località temporale* dice che:

- ☐ Se la CPU accede ad una locazione di memoria, è probabile che acceda presto a locazioni ad essa vicine.
- ☐ E' molto probabile che una CPU RISC acceda alla memoria in istanti temporali ben definiti.
- ☐ Nessuna delle altre risposte.
- ■ Se la CPU accede ad una locazione di memoria, è probabile che ri-acceda presto alla stessa locazione.
- ☐ La probabilità di accedere ad una locazione di memoria è inversamente proporzionale alla sua importanza.

Domanda 11 I *simboli non definiti* contenuti in un file oggetto (`.o`) non eseguibile:

- ■ Non sono associati ad indirizzi di memoria ne' assoluti ne' relativi; l'associazione ad un indirizzo di memoria avverrà solo nella fase di linking.
- ☐ Nei file `.o` non esiste alcuna nozione di "simboli", definiti o no.
- ☐ Sono associati ad indirizzi di memoria *assoluti*.
- ☐ Sono associati ad indirizzi di memoria *relativi*.
- ☐ Nessuna delle altre risposte.

Domanda 12

Si consideri la seguente funzione nel linguaggio C chiamata "sort" il cui scopo e' quello di ordinare un array in ingresso. Tale funzione prende in ingresso un array v[] (espresso naturalmente come puntatore a long long int) e la lunghezza n del vettore.
Al suo interno la funzione esegue una chiamata ad un'altra funzione denominata "swap" che scambia il valore dell'elemento del vettore in ingresso in posizione k con l'elemento successivo k+1. Quale delle implementazioni in assembly RISC-V della funzione swap e' corretta tra quelle proposte?

```
void sort (long long int v[], long long int n){
  long long int i, j;
  for (i=0; i<n; i+=1) {
   for (j=i-1; j>=0 && v[j] > v[j+1]; j-=1) {
    swap (v,j);
   }
  }
}

void swap (long long int v[], long long int k){
  long long int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
 }
 }
```

```
sort:  ...
       ...
       ...
       ...
       jal  swap
       ...
       ...
       ...

swap:  ...
       ...
       ...
       ...
       ...
       ...
       ...
```

■
```
swap:  slli  x7, x11, 3
       add   x7, x10, x7
       ld    x5, 0(x7)
       ld    x6, 8(x7)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       jalr  x0, 0(x1)
```

□
```
swap:  slli  x7, x11, 4
       add   x7, x10, x7
       ld    x7, 8(x7)
       sd    x5, 0(x7)
       jalr  x0, 0(x1)
```

□
```
swap:  slli  x7, x11, 3
       ld    x5, 0(x10)
       ld    x6, 8(x10)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       add   x7, x10, x7
```

□
```
swap:  srli  x7, x11, 3
       add   x7, x10, x7
       ld    x5, 0(x7)
       ld    x6, 8(x7)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       jalr  x0, 0(x1)
```

□ Nessuna delle altre risposte

Rispondere alle domande a risposta multipla annerendo la casella corrispondente alla risposta corretta. Ogni domanda ha una ed una sola risposta corretta.

Cognome e Nome: Nome32 Cognome32

Numero di Matricola: 22

Domanda 1 Svolgere in complemento a 2 su 4 bit l'operazione $0011_2 + 6_{10}$

- ☐ 0110_2
- ☐ 0111_2
- ■ Il risultato non è rappresentabile su 4 bit in CA2 (causa overflow)
- ☐ Nessuna delle altre risposte
- ☐ 1001_2

Domanda 2 Quale delle seguenti affermazioni è FALSA?

- ☐ Per essere eseguito, un programma Assembly deve essere tradotto in linguaggio macchina da un compilatore.
- ■ Una caratteristica dei programma scritti in linguaggio Assembly è la sua portabilità.
- ☐ Al livelo più basso, la CPU può capire solo programmi scritti in linguaggio macchina.
- ☐ Il linguaggio Assembly è strettamente legato alla CPU su cui il programma dovrà eseguire.
- ☐ Il linguaggio Assembly codifica le istruzioni macchina tramite codici mnemonici.

Domanda 3 Si consideri una CPU in cui le 5 fasi di esecuzione di un'istruzione impiegano $100ps$, $400ps$, $600ps$, $300ps$ e $100ps$. L'incremento di prestazioni che ci si può attendere usando una pipeline è:

- ☐ Nessuna delle altre risposte
- ☐ di 2 volte
- ☐ di 3.5 volte
- ☐ di 3 volte
- ■ di 2.5 volte

Domanda 4

```
int mathforfun(int i, int j,
               int q, int s) {
  for(i=0;i<q; i++)
  {
     q=q+s;
     q=q-j;
  }
  return q;
}
```

La funzione mathforfun prende in ingresso quattro argomenti e al suo interno svolge con questi delle operazioni matematiche ritornando un intero. Data la traduzione parziale in assembly intel qui sotto come completereste le righe X1 e X2 mancanti? scegliere una delle opzioni

```
mathforfun:
            pushq   %rbp
            movq    %rsp, %rbp
            movl    %edi, -4(%rbp)
            movl    %esi, -8(%rbp)
            movl    %edx, -12(%rbp)
            movl    %ecx, -16(%rbp)
            movl    $0, -4(%rbp)
.L3:
            movl    -4(%rbp), %eax
            X1
            jge     .L2
            movl    -16(%rbp), %eax
            addl    %eax, -12(%rbp)
            movl    -8(%rbp), %eax
            subl    %eax, -12(%rbp)
            X2
            jmp     .L3
.L2:
            movl    -12(%rbp), %eax
            popq    %rbp
            ret
```

- ☐ X1: `addl` $1, -4(%rbp)
  X2: `cmpl` -12(%rcx), %eax
- ■ X1: `cmpl` -12(%rbp), %eax
  X2: `addl` $1, -4(%rbp)
- ☐ X1: `addl` $1, -8(%rbp)
  X2: `cmpl` -12(%rbp), %eax
- ☐ X1: `movl` $1, -8(%rbp)
  X2: `cmpl` -10(%rbp), %eax
- ☐ Nessuna delle altre risposte

Domanda 5 Si consideri una fully associative grande $16KB$, con blocchi di 64 byte per blocco. In che blocco di cache è mappata la parola che sta all'indirizzo $0x100620$?

- ☐ Nel blocco numero 32.
- ☐ Nel blocco numero 0 o nel blocco numero 1 o nel blocco numero 2 o nel blocco numero 3.
- ☐ Nel blocco numero 24.
- ☐ Nel blocco numero 48 o nel blocco numero 49.
- ■ Nessuna delle altre risposte.

Domanda 6 Consideriamo la seguente combinazione di istruzioni: Tipo R: 25%, Tipo I (non load) 24%, Load 26%, Store 13%, Branch 9%, e Jump 3%. 1) Quale percentuale di tutte le istruzioni fa uso della memoria dati? 2) Quale percentuale delle istruzioni fa uso della memoria istruzioni? 3) Quale percentuale di tutte le istruzioni fa uso dell'estensione del segno? 4) Cosa fa l'estensione del segno nei cicli in cui il suo output non serve?

- ☐ 1) 35%, 2) 100%, 3) 76%, 4) L'estensione del segno produce un output solo nei cicli in cui necessario. Nei cicli in cui non è necessario, il circuito è disattivato.
- ☐ Nessuna delle altre risposte.
- ☐ 1) 37%, 2) 37%, 3) 76%, 4) L'estensione del segno produce un output ad ogni ciclo. Se l'output non è necessario viene semplicemente ignorato.
- ■ 1) 39%, 2) 100%, 3) 75%, 4) L'estensione del segno produce un output ad ogni ciclo. Se l'output non è necessario viene semplicemente ignorato.
- ☐ 1) 35%, 2) 100%, 3) 37%, 4) L'estensione del segno produce un output ad ogni ciclo. Se l'output non è necessario viene semplicemente ignorato.

Domanda 7 Usando la rappresentazione binaria, svolgere la somma $199 + 243$

- ☐ $199_{10} + 243_{10} = 111111010_2$
- ☐ $199_{10} + 243_{10} = 110111100_2$
- ☐ Nessuna delle altre risposte
- ☐ $199_{10} + 243_{10} = 110010010_2$
- ■ $199_{10} + 243_{10} = 110111010_2$

Domanda 8 Indicare l'esatto corrispondente in binario di 728_{10}

- ☐ Nessuna delle altre risposte
- ■ 001011011000_2
- ☐ 000111011000_2
- ☐ 000001101101_2
- ☐ 000001101110_2

Domanda 9 Inizialmente il contenuto di `x5` = 0x0000000000000000 Quale valore conterrá `x5` dopo aver eseguito le seguenti istruzioni Assembly RISC-V?

```
addi x6, x0, 0x000F
slli x6, x6, 28
or x5, x5, x6
```

- ☐ `x5` = 0000 1111 0000 0000 0000 0000 0000 1111 1111 0000 0000 0000 0000 0000 0000 0000_2
- ☐ `x5` = 1111 0000 0000 0000 0000 0000 0000 0000 1111 1111 0000 0000 0000 0000 0000 0000_2
- ■ `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 1111 0000 0000 0000 0000 0000 0000 0000_2
- ☐ `x5` = 1111 0000 0000 0000 0000 0000 0000 0000 1111 0000 0000 0000 0000 0000 0000 1111_2
- ☐ Nessuna delle altre risposte

Domanda 10 Quali delle seguenti singole istruzioni assembly RISC-V equivale alle due istruzioni

```
add x6, x5, x10
ld x7, 0(x6)
```

- ☐ `add x6, x7, x5`
- ■ Nessuna delle altre risposte
- ☐ `mv x6, x7`
- ☐ Tutte le risposte si equivalgono
- ☐ `ld x6, x7(x5)`

Domanda 11 Si dica a quale delle seguenti alternative corrisponde la seguente istruzione in assembly ARM:

```
mov  r7, r5, lsl #2
```

- ■ `r7 = 4 * r5;`
- ☐ `r5 = 2 * r7;`
- ☐ `r5 = 4 * r7;`
- ☐ `r7 = 2 * r5;`
- ☐ Nessuna delle altre risposte

Domanda 12

Si consideri la seguente funzione nel linguaggio C chiamata "sort" il cui scopo e' quello di ordinare un array in ingresso. Tale funzione prende in ingresso un array v[] (espresso naturalmente come puntatore a long long int) e la lunghezza n del vettore.
Al suo interno la funzione esegue una chiamata ad un'altra funzione denominata "swap" che scambia il valore dell'elemento del vettore in ingresso in posizione k con l'elemento successivo k+1. Quale delle implementazioni in assembly RISC-V della funzione swap e' corretta tra quelle proposte?

```
void sort (long long int v[], long long int n){
  long long int i, j;
  for (i=0; i<n; i+=1) {
   for (j=i-1; j>=0 && v[j] > v[j+1]; j-=1) {
    swap (v,j);
   }
  }
}

void swap (long long int v[], long long int k){
  long long int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
 }
 }
```

```
sort:  …
       …
       …
       …
       jal  swap
       …
       …
       …

swap:  …
       …
       …
       …
       …
       …
       …
       …
```

☐
```
swap:  srli  x7, x11, 3
       add   x7, x10, x7
       ld    x5, 0(x7)
       ld    x6, 8(x7)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       jalr  x0, 0(x1)
```

■
```
swap:  slli  x7, x11, 3
       add   x7, x10, x7
       ld    x5, 0(x7)
       ld    x6, 8(x7)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       jalr  x0, 0(x1)
```

☐
```
swap:  slli  x7, x11, 4
       add   x7, x10, x7
       ld    x7, 8(x7)
       sd    x5, 0(x7)
       jalr  x0, 0(x1)
```

☐
```
swap:  slli  x7, x11, 3
       ld    x5, 0(x10)
       ld    x6, 8(x10)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       add   x7, x10, x7
```

☐ Nessuna delle altre risposte

Rispondere alle domande a risposta multipla annerendo la casella corrispondente alla risposta corretta. Ogni domanda ha una ed una sola risposta corretta.

Cognome e Nome: `Nome43 Cognome43`

Numero di Matricola: `33`

Domanda 1 Usando la rappresentazione binaria, svolgere la somma $199 + 243$

- ☐ Nessuna delle altre risposte
- ☐ $199_{10} + 243_{10} = 111111010_2$
- ☐ $199_{10} + 243_{10} = 110010010_2$
- ☐ $199_{10} + 243_{10} = 110111100_2$
- ■ $199_{10} + 243_{10} = 110111010_2$

Domanda 2 Indicare l'esatto corrispondente in binario di 65535_{10}

- ☐ Nessuna delle altre risposte
- ■ 1111111111111111_2
- ☐ 1000000000000000_2
- ☐ 1100110011_2
- ☐ 1111000000000000_2

Domanda 3 Quali delle seguenti singole istruzioni assembly RISC-V equivale alle due istruzioni

```
add x6, x5, x10
ld x7, 0(x6)
```

- ☐ `mv x6, x7`
- ☐ Tutte le risposte si equivalgono
- ☐ `add x6, x7, x5`
- ■ Nessuna delle altre risposte
- ☐ `ld x6, x7(x5)`

Domanda 4 Inizialmente il contenuto di `x5` = 0x0000000000000000 Quale valore conterrá `x5` dopo aver eseguito le seguenti istruzioni Assembly RISC-V?

```
addi x6, x0, 0x000F
slli x6, x6, 28
or x5, x5, x6
```

- ☐ Nessuna delle altre risposte
- ■ `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 1111 0000 0000 0000 0000 0000 0000 0000_2
- ☐ `x5` = 1111 0000 0000 0000 0000 0000 0000 0000 1111 0000 0000 0000 0000 0000 0000 1111_2
- ☐ `x5` = 0000 1111 0000 0000 0000 0000 0000 1111 1111 0000 0000 0000 0000 0000 0000 0000_2
- ☐ `x5` = 1111 0000 0000 0000 0000 0000 0000 0000 1111 1111 0000 0000 0000 0000 0000 0000_2

Domanda 5 Svolgere la somma dei due numeri espressi in complemento a 2 su 12 bit $0111\,1011\,0101_2 + 1100\,0100\,1010_2$

- ☐ -1023_{10}
- ☐ -5119_{10}
- ☐ Nessuna delle altre risposte
- ☐ 5119_{10}
- ■ 1023_{10}

Domanda 6 Si consideri una cache direct mapped grande $16KB$, con blocchi di 64 byte per blocco. In che blocco di cache è mappata la parola che sta in memoria all'indirizzo $0x100400$?

- [x] Nel blocco numero 16.
- [ ] Nel blocco numero 32 o nel blocco numero 33.
- [ ] Nel blocco numero 0.
- [ ] Nel primo blocco libero.
- [ ] Nessuna delle altre risposte.

Domanda 7 Quale delle seguenti alternative è corretta, se si considera il seguente segmento di codice in assembly ARM:

```
        sub  r0, r0, #4
        mov  r3, #0
loop:   str  r3, [r0, #4]!
        cmp  r3, #10
        bne  loop
```

- [ ] Scrive i numeri da 1 a 10 nei primi 10 elementi dell'array di interi puntato da `r0`
- [x] Entra in un ciclo infinito
- [ ] E' scorretto sintatticamente
- [ ] Nessuna delle altre risposte
- [ ] Scrive i numeri da 0 a 9 nei primi 10 elementi dell'array di interi puntato da `r0`

Domanda 8 Consideriamo la seguente combinazione di istruzioni: Tipo R: 25%, Tipo I (non load) 24%, Load 26%, Store 13%, Branch 9%, e Jump 3%. 1) Quale percentuale di tutte le istruzioni fa uso della memoria dati? 2) Quale percentuale delle istruzioni fa uso della memoria istruzioni? 3) Quale percentuale di tutte le istruzioni fa uso dell'estensione del segno? 4) Cosa fa l'estensione del segno nei cicli in cui il suo output non serve?

- [ ] 1) 35%, 2) 100%, 3) 37%, 4) L'estensione del segno produce un output ad ogni ciclo. Se l'output non è necessario viene semplicemente ignorato.
- [ ] 1) 37%, 2) 37%, 3) 76%, 4) L'estensione del segno produce un output ad ogni ciclo. Se l'output non è necessario viene semplicemente ignorato.
- [ ] Nessuna delle altre risposte.
- [ ] 1) 35%, 2) 100%, 3) 76%, 4) L'estensione del segno produce un output solo nei cicli in cui necessario. Nei cicli in cui non è necessario, il circuito è disattivato.
- [x] 1) 39%, 2) 100%, 3) 75%, 4) L'estensione del segno produce un output ad ogni ciclo. Se l'output non è necessario viene semplicemente ignorato.

Domanda 9

```
int fattoriale(int n){
  if(n<=0) return 1;
  return n*fattoriale(n-1);
}
```

La funzione ricorsiva fattoriale prende in ingresso un intero e ne restituisce il fattoriale. Data la traduzione parziale in assembly intel qui sotto come completereste le righe X1 e X2 mancanti? scegliere una delle opzioni proposte

```
fattoriale(int):
                    pushq   %rbp
                    movq    %rsp, %rbp
                    subq    $16, %rsp
                    movl    %edi, -4(%rbp)
                    cmpl    $0, -4(%rbp)
                    jg      .L2
                    movl    $1, %eax
                    X1
.L2:
                    movl    -4(%rbp), %eax
                    X2
                    movl    %eax, %edi
                    call    fattoriale(int)
                    imull   -4(%rbp), %eax
.L3:
                    leave
                    ret
```

- [x] X1: `jmp` .L3
  X2: `subl` $1, %eax
- [ ] X1: `addl` $1, -4(%rbp)
  X2: `cmpl` -12(%rcx), %eax
- [ ] X1: `subl` $1, %eax
  X2: `cmpl` -10(%rbp), %eax
- [ ] Nessuna delle altre risposte
- [ ] X1: `addl` $1, -8(%rbp)
  X2: `jmp` .L2

Domanda 10 Le seguenti affermazioni descrivono alcuni dei pregi introdotti dalla pipeline nei microprocessori. Individua quale di queste NON è corretta.
Nelle architetture pipelined...

- [x] ...non si verificano mai eventi di hazard
- [ ] ...la frequenza di clock è determinata dall'istruzione più lenta
- [ ] ...l'ordine con cui sono scritte le istruzioni potrebbe influire sul tempo di esecuzione
- [ ] ...maggiore il numero di stadi, potenzialmente maggiori le prestazioni in confronto ad un'architettura senza pipeline
- [ ] Nessuna delle altre risposte

Domanda 11 I *simboli non definiti* contenuti in un file oggetto (`.o`) non eseguibile:

- [x] Non sono associati ad indirizzi di memoria ne' assoluti ne' relativi; l'associazione ad un indirizzo di memoria avverrà solo nella fase di linking.
- [ ] Nessuna delle altre risposte.
- [ ] Nei file `.o` non esiste alcuna nozione di "simboli", definiti o no.
- [ ] Sono associati ad indirizzi di memoria *relativi*.
- [ ] Sono associati ad indirizzi di memoria *assoluti*.

Domanda 12

Si consideri la seguente funzione nel linguaggio C chiamata "sort" il cui scopo e' quello di ordinare un array in ingresso. Tale funzione prende in ingresso un array v[] (espresso naturalmente come puntatore a long long int) e la lunghezza n del vettore.
Al suo interno la funzione esegue una chiamata ad un'altra funzione denominata "swap" che scambia il valore dell'elemento del vettore in ingresso in posizione k con l'elemento successivo k+1. Quale delle implementazioni in assembly RISC-V della funzione swap e' corretta tra quelle proposte?

```
void sort (long long int v[], long long int n){
  long long int i, j;
  for (i=0; i<n; i+=1) {
   for (j=i-1; j>=0 && v[j] > v[j+1]; j-=1) {
    swap (v,j);
   }
  }
}

void swap (long long int v[], long long int k){
  long long int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
 }
 }
```

```
sort:  …
       …
       …
       …
       jal  swap
       …
       …
       …

swap:  …
       …
       …
       …
       …
       …
       …
       …
```

☐ Nessuna delle altre risposte

☐
```
swap:  slli  x7, x11, 3
       ld    x5, 0(x10)
       ld    x6, 8(x10)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       add   x7, x10, x7
```

☐
```
swap:  slli  x7, x11, 4
       add   x7, x10, x7
       ld    x7, 8(x7)
       sd    x5, 0(x7)
       jalr  x0, 0(x1)
```

☐
```
swap:  srli  x7, x11, 3
       add   x7, x10, x7
       ld    x5, 0(x7)
       ld    x6, 8(x7)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       jalr  x0, 0(x1)
```

■
```
swap:  slli  x7, x11, 3
       add   x7, x10, x7
       ld    x5, 0(x7)
       ld    x6, 8(x7)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       jalr  x0, 0(x1)
```

Rispondere alle domande a risposta multipla annerendo la casella corrispondente alla risposta corretta. Ogni domanda ha una ed una sola risposta corretta.

Cognome e Nome: `Nome54 Cognome64`

Numero di Matricola: `44`

Domanda 1 Usando la rappresentazione binaria, svolgere la somma $183 + 37$

- ☐ $183_{10} + 37_{10} = 11011001_2$
- ☐ Nessuna delle altre risposte
- ■ $183_{10} + 37_{10} = 11011100_2$
- ☐ $183_{10} + 37_{10} = 11001101_2$
- ☐ $183_{10} + 37_{10} = 101011100_2$

Domanda 2 Come si rappresenta in decimale il numero binario 101101101100_2?

- ■ 2924_{10}
- ☐ 731_{10}
- ☐ 877_{10}
- ☐ 5848_{10}
- ☐ Nessuna delle altre risposte

Domanda 3 Svolgere la somma dei due numeri espressi in complemento a 2 su 4 bit $0101_2 + 1010_2$

- ☐ 15_{10}
- ☐ -15_{10}
- ■ -1_{10}
- ☐ Nessuna delle altre risposte
- ☐ 1111_{10}

Domanda 4 Inizialmente il contenuto di `x5` = 0x0000000000000000 Quale valore conterrá `x5` dopo aver eseguito le seguenti istruzioni Assembly RISC-V?

```
addi x6, x0, 0x000F
slli x6, x6, 28
or x5, x5, x6
```

- ■ `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 1111 0000 0000 0000 0000 0000 0000 0000_2
- ☐ `x5` = 1111 0000 0000 0000 0000 0000 0000 0000 1111 0000 0000 0000 0000 0000 0000 1111_2
- ☐ `x5` = 1111 0000 0000 0000 0000 0000 0000 0000 1111 1111 0000 0000 0000 0000 0000 0000_2
- ☐ Nessuna delle altre risposte
- ☐ `x5` = 0000 1111 0000 0000 0000 0000 0000 1111 1111 0000 0000 0000 0000 0000 0000 0000_2

Domanda 5 Quali delle seguenti singole istruzioni assembly RISC-V equivale alle due istruzioni

```
add x6, x5, x10
ld x7, 0(x6)
```

- ☐ `add x6, x7, x5`
- ☐ Tutte le risposte si equivalgono
- ☐ `mv x6, x7`
- ☐ `ld x6, x7(x5)`
- ■ Nessuna delle altre risposte

Domanda 6 Si dica a quale delle seguenti alternative corrisponde la seguente istruzione in assembly ARM:

```
add  r0, r1, r1, lsl #1
```

- [ ] `r0 = r0 * r1;`
- [ ] Nessuna delle altre risposte
- [ ] `r0 = r0 + (2 * r1);`
- [x] `r0 = 3 * r1;`
- [ ] `r1 = r0 + (2 * r1);`

Domanda 7 Supponiamo di essere in un ciclo di clock in cui il processore RISC-V carica dalla memoria istruzioni la seguente parola: `00000000001100010000000010110011`. 1) Quale è l'istruzione codificata. 2) Quale è il valore degli ingressi dell'unità di controllo della ALU per questa istruzione? 3) Quale è il nuovo indirizzo del PC dopo che l'istruzione è stata eseguita?

- [ ] 1) L'istruzione codificata è `add x2, x1, x3`. 2) `ALUop` = 10, `ALU Control Lines` = 0010. 3) PC = PC + 4 se X12 diverso da X13, PC = PC + 20*4 se X12 è uguale a X13.
- [x] 1) L'istruzione codificata è `add x1, x2, x3`. 2) `ALUop` = 10, `ALU Control Lines` = 0010. 3) PC = PC + 4.
- [ ] Nessuna delle altre risposte.
- [ ] 1) L'istruzione codificata è `sub x1, x2, x3`. 2) `ALUop` = 00, `ALU Control Lines` = 0110. 3) PC = PC + 4.
- [ ] 1) L'istruzione codificata è `sub x1, x2, x3`. 2) `ALUop` = 10, `ALU Control Lines` = 0110. 3) PC = PC + 4.

Domanda 8 Si consideri una cache direct mapped grande $64KB$, con blocchi di 64 byte per blocco. In che blocco di cache è mappata la parola che sta in memoria all'indirizzo $0x1F040$?

- [ ] Nel primo blocco libero.
- [ ] Nessuna delle altre risposte.
- [ ] Non si può dire senza conoscere la dimensione della memoria principale.
- [x] Nel blocco numero 961.
- [ ] Nel blocco numero 40.

Domanda 9 Si consideri una CPU che impiega $600ps$ per la fase di fetch, $600ps$ per la fase di decodifica, $500ps$ per eseguire operazioni con la ALU, $400ps$ per la fase di accesso alla memoria e $700ps$ per la fase di scrittura nel register file. Il massimo incremento di prestazioni che ci si può attendere usando una pipeline è:

- [ ] di 3 volte
- [ ] di 2.5 volte
- [ ] di 2 volte
- [x] di 4 volte
- [ ] Nessuna delle altre risposte

Domanda 10 Le *librerie statiche*:

- [ ] Vengono utilizzate dall'Assembler per implementare le macro/pseudo-istruzioni.
- [ ] Nessuna delle altre risposte.
- [ ] Sono effettivamente collegate al programma solo quando esso viene caricato (in caso di linking *non lazy*) o eseguito (in caso di *lazy linking*).
- [ ] Possono essere utilizzate da programmi C, ma non da programmi scritti in Assembly.
- [x] Sono utilizzate in fase di linking, ma non servono per il caricamento o l'esecuzione dell'eseguibile finale.

Domanda 11

```
int mathforfun(int i, int j,
                int q, int s) {
  for(i=0;i<q; i++)
  {
     q=q+s;
     q=q-j;
  }
  return q;
}
```

La funzione mathforfun prende in ingresso quattro argomenti e al suo interno svolge con questi delle operazioni matematiche ritornando un intero. Data la traduzione parziale in assembly intel qui sotto come completereste le righe X1 e X2 mancanti? scegliere una delle opzioni

```
 mathforfun:
              pushq   %rbp
              movq    %rsp, %rbp
              movl    %edi, -4(%rbp)
              movl    %esi, -8(%rbp)
              movl    %edx, -12(%rbp)
              movl    %ecx, -16(%rbp)
              movl    $0, -4(%rbp)
 .L3:
              movl    -4(%rbp), %eax
              X1
              jge     .L2
              movl    -16(%rbp), %eax
              addl    %eax, -12(%rbp)
              movl    -8(%rbp), %eax
              subl    %eax, -12(%rbp)
              X2
              jmp     .L3
 .L2:
              movl    -12(%rbp), %eax
              popq    %rbp
              ret
```

☐ Nessuna delle altre risposte

☐ X1: `addl` $1, -8(%rbp)
X2: `cmpl` -12(%rbp), %eax

■ X1: `cmpl` -12(%rbp), %eax
X2: `addl` $1, -4(%rbp)

☐ X1: `addl` $1, -4(%rbp)
X2: `cmpl` -12(%rcx), %eax

☐ X1: `movl` $1, -8(%rbp)
X2: `cmpl` -10(%rbp), %eax

Domanda 12

Si consideri la seguente funzione nel linguaggio C chiamata ”sort” il cui scopo e’ quello di ordinare un array in ingresso. Tale funzione prende in ingresso un array v[] (espresso naturalmente come puntatore a long long int) e la lunghezza n del vettore.
Al suo interno la funzione esegue una chiamata ad un’altra funzione denominata ”swap” che scambia il valore dell’elemento del vettore in ingresso in posizione k con l’elemento successivo k+1. Quale delle implementazioni in assembly RISC-V della funzione swap e’ corretta tra quelle proposte?

```
void sort (long long int v[], long long int n){
  long long int i, j;
  for (i=0; i<n; i+=1) {
   for (j=i-1; j>=0 && v[j] > v[j+1]; j-=1) {
    swap (v,j);
   }
  }
}

void swap (long long int v[], long long int k){
  long long int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
 }
 }
```

```
sort:  …
       …
       …
       …
       jal  swap
       …
       …
       …

swap:  …
       …
       …
       …
       …
       …
       …
       …
```

```
swap:  srli  x7, x11, 3
       add   x7, x10, x7
       ld    x5, 0(x7)
       ld    x6, 8(x7)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       jalr  x0, 0(x1)
```

```
swap:  slli  x7, x11, 4
       add   x7, x10, x7
       ld    x7, 8(x7)
       sd    x5, 0(x7)
       jalr  x0, 0(x1)
```

```
swap:  slli  x7, x11, 3
       ld    x5, 0(x10)
       ld    x6, 8(x10)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       add   x7, x10, x7
```

```
swap:  slli  x7, x11, 3
       add   x7, x10, x7
       ld    x5, 0(x7)
       ld    x6, 8(x7)
       sd    x6, 0(x7)
       sd    x5, 8(x7)
       jalr  x0, 0(x1)
```

☐ Nessuna delle altre risposte

Rispondere alle domande a risposta multipla annerendo la casella corrispondente alla risposta corretta. Ogni domanda ha una ed una sola risposta corretta.

Cognome e Nome: Nome65 Cognome65

Numero di Matricola: 55

Domanda 1 Usando la rappresentazione binaria, svolgere la sottrazione $10110011101100011001 - 0x4AA95$

- [ ] 011000010000100101000_2
- [ ] Nessuna delle altre risposte
- [x] 011010010000100001000_2
- [ ] 101110011000000010000_2
- [ ] 101100010001000010000_2

Domanda 2 Si consideri una cache direct mapped grande $4KB$, con blocchi di 16 byte per blocco. In che blocco di cache è mappata la parola che sta in memoria all'indirizzo $0x1F164$?

- [ ] Nel blocco numero 64.
- [x] Nel blocco numero 22.
- [ ] Nessuna delle altre risposte.
- [ ] Nel blocco numero 6.
- [ ] Nel primo blocco libero.

Domanda 3 Considerare l'operazione di somma tra interi in assembly RISC-V. Essa puó essere richiesta:

- [ ] Con due operandi sorgenti registri e un operando destinazione in memoria
- [ ] Con due operandi sorgenti registri e una destinazione anche esso registro
- [x] Con due operandi sorgenti di tipo registro e un operando destinazione di tipo registro, oppure con un operando di tipo registro e un operando immediato come sorgenti e un operando registro destinazione
- [ ] Con due operandi sorgenti immediati e un operando destinazione registro
- [ ] Nessuna delle altre risposte

Domanda 4 Le *librerie statiche*:

- [ ] Possono essere utilizzate da programmi C, ma non da programmi scritti in Assembly.
- [ ] Nessuna delle altre risposte.
- [ ] Vengono utilizzate dall'Assembler per implementare le macro/pseudo-istruzioni.
- [ ] Sono effettivamente collegate al programma solo quando esso viene caricato (in caso di linking *non lazy*) o eseguito (in caso di *lazy linking*).
- [x] Sono utilizzate in fase di linking, ma non servono per il caricamento o l'esecuzione dell'eseguibile finale.

Domanda 5 Consideriamo la seguente combinazione di istruzioni: Tipo R: 25%, Tipo I (non load) 24%, Load 26%, Store 13%, Branch 9%, e Jump 3%. 1) Quale percentuale di tutte le istruzioni fa uso della memoria dati? 2) Quale percentuale delle istruzioni fa uso della memoria istruzioni? 3) Quale percentuale di tutte le istruzioni fa uso dell'estensione del segno? 4) Cosa fa l'estensione del segno nei cicli in cui il suo output non serve?

- [x] 1) 39%, 2) 100%, 3) 75%, 4) L'estensione del segno produce un output ad ogni ciclo. Se l'output non è necessario viene semplicemente ignorato.
- [ ] Nessuna delle altre risposte.
- [ ] 1) 35%, 2) 100%, 3) 37%, 4) L'estensione del segno produce un output ad ogni ciclo. Se l'output non è necessario viene semplicemente ignorato.

☐ 1) 37%, 2) 37%, 3) 76%, 4) L'estensione del segno produce un output ad ogni ciclo. Se l'output non è necessario viene semplicemente ignorato.

☐ 1) 35%, 2) 100%, 3) 76%, 4) L'estensione del segno produce un output solo nei cicli in cui necessario. Nei cicli in cui non è necessario, il circuito è disattivato.

Domanda 6 A quale numero decimale corrisponde la seguente cifra binaria codificata secondo lo standard IEEE754?

s	e	m
1	10000101	11101101010000000000000

■ Corrisponde al decimale -123.3125

☐ Nessuna delle altre risposte

☐ Corrisponde al decimale $-1.926757813 \times 2^{139}$

☐ Corrisponde al decimale -118.625

☐ Corrisponde al decimale 118.625

Domanda 7 Svolgere in complemento a 2 su 4 bit l'operazione $0011_2 + 6_{10}$

☐ Nessuna delle altre risposte

☐ 0110_2

☐ 0111_2

■ Il risultato non è rappresentabile su 4 bit in CA2 (causa overflow)

☐ 1001_2

Domanda 8

```
int mathforfun(int i, int j,
               int q, int s) {
  for(i=0;i<q; i++)
  {
     q=q+s;
     q=q-j;
  }
  return q;
}
```

La funzione mathforfun prende in ingresso quattro argomenti e al suo interno svolge con questi delle operazioni matematiche ritornando un intero. Data la traduzione parziale in assembly intel qui sotto come completereste le righe X1 e X2 mancanti? scegliere una delle opzioni

```
mathforfun:
                pushq   %rbp
                movq    %rsp, %rbp
                movl    %edi, -4(%rbp)
                movl    %esi, -8(%rbp)
                movl    %edx, -12(%rbp)
                movl    %ecx, -16(%rbp)
                movl    $0, -4(%rbp)
.L3:
                movl    -4(%rbp), %eax
                X1
                jge     .L2
                movl    -16(%rbp), %eax
                addl    %eax, -12(%rbp)
                movl    -8(%rbp), %eax
                subl    %eax, -12(%rbp)
                X2
                jmp     .L3
.L2:
                movl    -12(%rbp), %eax
                popq    %rbp
                ret
```

☐ X1: `addl` $1, -4(%rbp)
X2: `cmpl` -12(%rcx), %eax

☐ X1: `addl` $1, -8(%rbp)
X2: `cmpl` -12(%rbp), %eax

■ X1: `cmpl` -12(%rbp), %eax
X2: `addl` $1, -4(%rbp)

☐ X1: `movl` $1, -8(%rbp)
X2: `cmpl` -10(%rbp), %eax

☐ Nessuna delle altre risposte

Domanda 9 Le seguenti affermazioni descrivono alcuni dei pregi introdotti dalla pipeline nei microprocessori. Individua quale di queste NON è corretta.
Nelle architetture pipelined...

☐ ...l'ordine con cui sono scritte le istruzioni potrebbe influire sul tempo di esecuzione

■ ...non si verificano mai eventi di hazard

☐ ...la frequenza di clock è determinata dall'istruzione più lenta

☐ Nessuna delle altre risposte

☐ ...maggiore il numero di stadi, potenzialmente maggiori le prestazioni in confronto ad un'architettura senza pipeline

Domanda 10 Si dica a quale delle seguenti alternative corrisponde la seguente istruzione in assembly ARM:

```
add  r0, r1, r1, lsl #1
```

☐ `r1 = r0 + (2 * r1);`

☐ Nessuna delle altre risposte

■ `r0 = 3 * r1;`

☐ `r0 = r0 + (2 * r1);`

☐ `r0 = r0 * r1;`

Domanda 11 Il registro `x5` contiene il valore `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1100 1100 1111 1111 0011 0000. Quale valore conterrá `x5` dopo aver eseguito le seguenti istruzioni Assembly RISC-V?

```
srli x6, x5, 4
slli x5, x5, 10
and x5, x5, x6
```

■ `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1100 1100 0000 0000 0000_2

☐ Nessuna delle altre risposte

☐ `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 0011 0011 1111 1100 1100 0000 0000 0000_2

☐ `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 0001 1100_2

☐ `x5` = 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1100 1100 1111 1111 0011_2

Domanda 12 La funzione `sumV` prende in ingresso l'indirizzo di tre array di interi `u`, `v`, `z`, e la dimensione `size` degli array, il suo compito è quello di effettuare le operazioni sulla memoria specificate nel programma. Il compilatore fornisce la seguente traduzione in assembly RISC-V che è incompleta: le righe X1, X2 e X3 sono omesse. (Sono rispettate le convenzioni di chiamata per gli argomenti e per il valore di ritorno specificate dall'ABI vista durante il corso) Quale delle triple X1, X2 e X3 proposte corrisponde alle righe corrette?

```
typedef long long int int64;
typedef unsigned long long int uint64;
void sumV(int64 ** u, int64 ** v, int64 ** z, uint64 size){
 for (int64 r = 0; r < size; r++) {
  for (int64 c = 0; c < size; c++) {
   int64 p = 0;
   for (uint64 k = 0; k < size; k++) {
    p += *(*(v + r) + k) * *(*(u+k) + c);
   }
    *(*(v+r) + c) = p;
  }
 }
 return;
}
```

```
sumV:
        beqz    a3,.L1
        slli    t4,a3,3
        mv      t5,a1
        add     t6,a1,t4
.L3:
        X1
.L6:
        mv      a2,a0
        mv      a5,t3
        li      a1,0
.L4:
        X2
        ld      a4,0(a3)
        addi    a2,a2,8
        mul     a4,a4,a6
        add     a1,a1,a4
        X3
.L1:
        ret
```

```
X1
    ld t3,0(t5)
    li a7,0
    add t1,t4,t3
X2
    ld a3,0(a2)
    ld a6,0(a5)
    addi a5,a5,4
    add a3,a3,a7
X3
    bne t1,a5,.L4
    add a5,t3,a7
    sd a1,0(a5)
    addi a7,a7,8
    bne t4,a7,.L6
    addi t5,t5,8
    bne t6,t5,.L3
```

X1
```
ld t3,0(t5)
li a7,0
add t3,t4,t3
```
X2
```
ld a3,0(a2)
ld a6,0(a5)
addi a5,a5,8
```
☐ `add a3,a3,a7`

X3
```
bne t1,a5,.L4
add a5,t3,a7
sd a1,0(a5)
addi a7,a7,8
bne t4,a7,.L6
addi t5,t5,8
bne t6,t5,.L3
```

X1
```
ld t3,0(t5)
li a7,0
add t1,t4,t3
```
X2
```
ld a3,0(a2)
ld a6,0(a5)
addi a5,a5,8
```
☐ `add a3,a3,a7`

X3
```
beq t1,a5,.L4
add a5,t3,a7
sd a1,0(a5)
addi a7,a7,8
beq t4,a7,.L6
addi t5,t5,8
beq t6,t5,.L3
```

X1
```
ld t3,0(t5)
li a7,0
add t4,t1,t3
```
X2
```
ld a3,0(a2)
ld a6,0(a5)
addi a5,a5,8
```
☐ `add a7,a3,a7`

X3
```
bne t1,a5,.L4
add a5,t3,a7
sd a1,0(a5)
addi a7,a7,8
bne t4,a7,.L6
addi t5,t5,8
bne t6,t5,.L3
```

■ Nessuna delle altre risposte