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: `Nome1 Cognome1`

Numero di Matricola: 0

Domanda 1 Quali delle seguenti singole istruzioni assembly RISC-V equivale alle due istruzioni

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

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

Domanda 2 Indicare l'esatto corrispondente in binario di 728_{10}

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

Domanda 3 Svolgere in complemento a 2 su 8 bit l'operazione $1101_{10} - 125_{10}$

- ☐ $0111\,1111_2$
- ■ Nessuna delle altre risposte
- ☐ $1111\,1111_2$
- ☐ $1011\,1111_2$
- ☐ $1000\,0000_2$

Domanda 4 Le seguenti affermazioni descrivono alcuni dei pregi introdotti dalla pipeline nei microprocessori. Individua quale di queste NON è corretta.
Nelle architetture pipelined...

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

Domanda 5 Usando la rappresentazione binaria, svolgere la somma $8494 + 7726$

- ☐ $8494_{10} + 7726_{10} = 1100001111110101_2$
- ☐ $8494_{10} + 7726_{10} = 1111010111000011_2$
- ■ $8494_{10} + 7726_{10} = 0011111101011100_2$
- ☐ $8494_{10} + 7726_{10} = 0101110000111111_2$
- ☐ Nessuna delle altre risposte

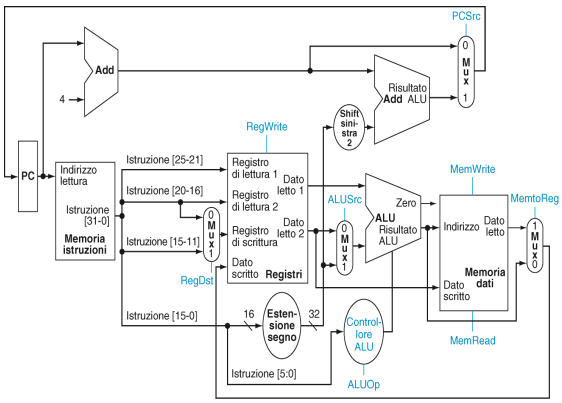

Figure 1: Schema di datapath

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

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

- ☐ Nessuna delle altre risposte
- ☐ `r0 = r0 + (2 * r1);`
- ☐ `r1 = r0 + (2 * r1);`
- ☐ `r0 = r0 * r1;`
- ■ `r0 = 3 * r1;`

Domanda 7 Si consideri lo schema in Figure 1. Si indichi quale delle seguenti alternative spiega meglio le modalità di esecuzione di un'istruzione di tipo `R`.

- ☐ Dopo la fase di prelievo, `RegDst` viene selezionato a 0, `ALUSrc` viene selezionato a 0, `RegWrite` viene posto a 1, `MemToReg` viene posto a 0, `AluOP` viene posto a 01
- ■ Dopo la fase di prelievo, `RegDst` viene selezionato a 1, `ALUSrc` viene selezionato a 0, `RegWrite` viene posto a 1, `MemToReg` viene posto a 0, `AluOP` viene posto a 10
- ☐ Nessuna delle scelte proposte.
- ☐ Dopo la fase di prelievo, `RegDst` viene selezionato a 1, `ALUSrc` viene selezionato a 1, `RegWrite` viene posto a 1, `MemToReg` viene posto a 0, `AluOP` viene posto a 11

Domanda 8 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 $0x100400$?

- ☐ Nel primo blocco libero.
- ☐ Nel blocco numero 32 o nel blocco numero 33.
- ☐ Nessuna delle altre risposte.
- ■ Nel blocco numero 16.
- ☐ Nel blocco numero 0.

Domanda 9

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

Domanda 10 Quale delle seguenti affermazioni è FALSA?

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

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` = $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1100\ 1100\ 1111\ 1111\ 0011_2$
- ■ `x5` = $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1100\ 1100\ 0000\ 0000\ 0000_2$
- ☐ Nessuna delle altre risposte
- ☐ `x5` = $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0011\ 0001\ 1100_2$
- ☐ `x5` = $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0011\ 0011\ 1111\ 1100\ 1100\ 0000\ 0000\ 0000_2$

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, 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)
```

■
```
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

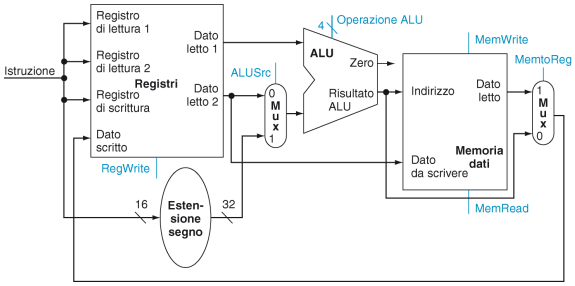

Figure 1: Schema di porzione del datapath.

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: `Nome2 Cognome2`

Numero di Matricola: `1`

Domanda 1 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
```

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

Domanda 2 Con riferimento alla Figura 1, si dica a cosa serve il blocco estensione segno.

- [ ] L'estensione è usata solo nell'eventualità di salti condizionati per individuare correttamente il registro da sommare al PC.
- [ ] Nessuna delle altre risposte
- [x] Gli operandi immediati sono codificati su 16 bit. Mentre i registri sono a 32, quindi occorre estendere l'operando per effettuare operazioni con il registro preservandone il segno.
- [ ] L'estensione occorre per individuare correttamente il registro destinatario.
- [ ] Il MIPS consente di utilizzare anche porzioni di registri a 16 bit e in questo caso bisogna estendere l'operando.

Domanda 3 Si dica a quale delle seguenti alternative corrisponde la seguente istruzione in assembly ARM:

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

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

Domanda 4 Riportare in binario il risultato della differenza $11000.1011 - 111.111101$

- [ ] $11000.1011 - 111.111101 = 100001011.11$
- [ ] $11000.1011 - 111.111101 = 1000.101101$
- [ ] $11000.1011 - 111.111101 = 11000.0101110$
- [x] $11000.1011 - 111.111101 = 10000.101111$
- [ ] Nessuna delle altre risposte

Domanda 5 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 $0x1F040$?

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

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

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

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

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

Domanda 8 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
- [x] di 4 volte
- [ ] Nessuna delle altre risposte
- [ ] di 2 volte
- [ ] di 2.5 volte

Domanda 9 Considerare l'operazione di somma tra interi in assembly RISC-V. Essa puó essere richiesta:

- [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 registri e un operando destinazione in memoria
- [ ] Con due operandi sorgenti registri e una destinazione anche esso registro
- [ ] Nessuna delle altre risposte
- [ ] Con due operandi sorgenti immediati e un operando destinazione registro

Domanda 10

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

Domanda 11 Quale delle seguenti affermazioni è FALSA, quando si parla dell'ISA (Instruction Set Architecture)?

- [ ] Definisce le diverse modalità di accesso alla memoria (indirizzamento).
- [ ] É l'insieme delle istruzioni macchina che la CPU riconosce e sa eseguire.
- [ ] Definisce il numero e tipo di registri della CPU.
- [x] Definisce la convenzioni di chiamata a subroutine (i.e., come e dove passare i parametri).
- [ ] Definisce la sintassi e la semantica delle istruzione macchina.

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, 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)
```

☐
```
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
```

+3/1/50+

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: Nome3 Cognome3

Numero di Matricola: 2

Domanda 1 Le *librerie statiche*:

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

Domanda 2 A quale numero decimale corrisponde la cifra esadecimale $0xE7E80000$ codificata secondo lo standard IEEE754?

- ☐ Corrisponde al decimale -1.5625000×2^{16}
- ☐ Corrisponde al decimale -1.5546875×2^{18}
- ☐ Corrisponde al decimale -1.8046875×2^{82}
- ■ Corrisponde al decimale -1.8125000×2^{80}
- ☐ Nessuna delle altre risposte

Domanda 3 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 primo blocco libero.
- ☐ Nel blocco numero 64.
- ☐ Nessuna delle altre risposte.
- ☐ Nel blocco numero 6.
- ■ Nel blocco numero 22.

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 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 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

Domanda 5 Per le istruzioni di tipo load word e store word:

- [ ] la ALU non calcola l'indirizzo di memoria
- [x] la ALU deve eseguire una somma per calcolare l'indirizzo di memoria
- [ ] Nessuna delle altre risposte
- [ ] la ALU esegue un'operazione AND per calcolare l'indirizzo di memoria
- [ ] la ALU deve eseguire una sottrazione per calcolare l'indirizzo di memoria

Domanda 6 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
```

- [x] Entra in un ciclo infinito
- [ ] Scrive i numeri da 1 a 10 nei primi 10 elementi dell'array di interi puntato da `r0`
- [ ] 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 7 Svolgere in complemento a 2 su 4 bit l'operazione $0011_2 + 6_{10}$

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

Domanda 8 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 volte
- [x] di 4 volte
- [ ] Nessuna delle altre risposte
- [ ] di 2.5 volte

Domanda 9 Usando la rappresentazione binaria, svolgere la somma $623 + 412$

- [ ] Nessuna delle altre risposte
- [x] $623_{10} + 412_{10} = 010000001011_2$
- [ ] $623_{10} + 412_{10} = 001111111011_2$
- [ ] $623_{10} + 412_{10} = 101100000100_2$
- [ ] $623_{10} + 412_{10} = 101111110011_2$

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

- ☐ Con due operandi sorgenti immediati e un operando destinazione registro
- ☐ Con due operandi sorgenti registri e una destinazione anche esso registro
- ■ 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
- ☐ Nessuna delle altre risposte
- ☐ Con due operandi sorgenti registri e un operando destinazione in memoria

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

☐
```
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