1) Descrivere la fase di *fetch* e quella di *execute* di un processore.

Fase di *fetch*:

a) *Fetch* del codice operativo: il Program Counter viene inviato verso la memoria - viene ordinata la lettura - il dato letto viene trasferito nell'Istruction Register - il P.C. viene incrementato;

I passi successivi dipendono dal *codice operativo* presente nell'I.R.

b) *Fetch* dell'operando: il Program Counter viene inviato verso la memoria - viene ordinata la lettura - il dato letto viene trasferito nei registri interni - il P.C. viene incrementato;

Oppure:

c) *Fetch* dell'indirizzo dell'operando: il Program Counter viene inviato verso la memoria - viene ordinata la lettura - il dato letto viene trasferito nei registri di indirizzo - il P.C. viene incrementato; l'operazione viene ripetuta se il trasferimento dell'indirizzo richiede più cicli;

Fase di *execute*: dipende dall'istruzione da eseguire. Ecco alcuni esempi:

a) Istruzione di MOVE da registro a memoria:
   i) Il dato (contenuto nel registro di dato nominato nell'istruzione) e l'indirizzo (precedentemente catturato nella fase di fetch) vengono inviati verso la memoria - viene ordinata la scrittura;
b) Istruzione di ADDizione:
   i) Si configura l'ALU per l'operazione richiesta (il risultato è subito disponibile sui fili di uscita dell'ALU);
   ii) Si comanda la memorizzazione nel registro di destinazione dell'operazione;
c) Istruzione di JUMP (salto incondizionato):
   i) Il contenuto del PC viene rimpiazzato con il contenuto del registro di indirizzo (precedentemente catturato nella fase di fetch): si comanda la memorizzazione del contenuto del registro di indirizzo nel PC.

2) Qual è il contenuto del *Program Counter* e quello dell'*Istruction Register* alla fine dell'esecuzione di un'istruzione?

Alla fine della fase di *execute* il contenuto del Program Counter può essere:

- quello alla fine dell'ultima fase di *fetch* (dopo l'autoincremento): quindi il PC punta all'istruzione successiva a quella appena eseguita, OVVERO
- quello modificato dall'istruzione di salto appena eseguita (vedi gli esempi di esecuzione della domanda precedente): in questo caso non c'è relazione spaziale con l'istruzione precedente.

Il Registro-Istruzioni invece contiene il codice operativo dell'*istruzione appena eseguita*.

3) Un microprocessore ha uno spazio di indirizzamento di 32 bit. Qual è la quantità di memoria direttamente indirizzabile?

32 bit $\Rightarrow$ Celle di memoria = 2^{32}
$2^{30} \rightarrow$1GB
$\Rightarrow$ Memoria $2^{32} = 2^2 x 2^{30} = 4$ GB di memoria indirizzabile

4) Conoscendo le caratteristiche di due processori, quali parametri bisognerebbe considerare per confrontare la velocità di esecuzione dei programmi?

Per effettuare un benchmark completo su due processori occorre innanzitutto tener presente i tipi di programmi con i quali si vuole effettuare il test: non tutte le CPU infatti sono ottimizzate per eseguire lo stesso tipo di operazione: ad esempio alcune sono ottimizzate per applicazioni più professionali di un certo settore, altre possono essere più avvantaggiate nell'elaborare dati per il rendering bidimensionale o tridimensionale.
In generale ciò che distingue maggiormente una CPU da un'altra è la frequenza di clock. Sarebbe tuttavia alquanto riduttivo limitarsi a tali valori, dato che vi sono molte variabili che possono condizionare la velocità di calcolo di un microprocessore: il tipo e la quantità della memoria cache, i tipi di bus e la loro frequenza di lavoro, ecc… Si considerino per tanto alcuni casi pratici.
Molti rivenditori hardware cercano accanitamente di ammaliare ogni potenziale compratore inesperto cercando di far leva sulla frequenza di lavoro dei processori. Ad esempio se si confrontano un processore "Pentium" con un "Celeron" a parità di frequenza,le prestazioni saranno molto differenti, stesso discorso per "Athlon" e "Duron" o "Via". Se poi, sempre a parità di frequenza, si considera un processore Mac con 2 MB di cache allora ci si può visibilmente accorgere della differenza abissale che separa i differenti costrutti di processori.
(Nei sistemi sui quali "girano" grossi applicativi sviluppati su più moduli residenti su disco e richiamati in memoria alla bisogna, ovvero quando si elaborano grossi moli di dati, diventa rilevante anche la velocità di accesso ai dischi!)

5) Su un Personal Computer un programma viene eseguito più velocemente se c'è più memoria?

Generalmente parlando, la risposta è No: la velocità di elaborazione dipende dalla velocità del microprocessore (e quindi dalla frequenza del cloack) e dalla velocità di risposta della memoria, non dalla dimensione della memoria.
Una grande quantità di memoria può facilitare l'esecuzione immediata (senza scrittura su HD di dati temporanei) nei casi in cui si utilizzi un O.S. "multi-tasking" (vedi windows o linux) o multi-utente.

Vale anche:
In genere un programma viene eseguito più velocemente se

- Vi è una maggiore quantità di memoria cache interna al processore(entro certi limiti): ad esempio la serie di processori ad alte prestazioni della Intel, la Xeon, ha una cache interna che è circa il quadruplo di quella normalmente presente nella serie Pentium.
- Vi è una maggior quantità di memoria Ram, in quanto essa permette un accesso al disco fisso(che è più lento sia in lettura che nel tempo di accesso rispetto alla memoria ad accesso casuale) più raro. Questa considerazione vale finché la Ram non contiene tutto il programma, dal momento in cui essa è più grande del programma, un successivo aumento della sua dimensione non influenza le prestazioni del sistema (considerando nulla l'influenza dell'OS su cui gira il programma)

6) Possedendo un PC e volendo espandere la memoria principale (RAM), in base a quali considerazioni si può decidere sulla quantità massima di memoria che ha senso mettere nel PC?

Innanzitutto occorre sapere quanta memoria è gestibile dal sistema (larghezza dei registri di indirizzo della CPU, come il Program Counter ed eventuali altri, larghezza dell'ADDRESS BUS del sistema, ecc.), successivamente è necessario considerare l'utilizzo che si intende fare del PC, come già precisato in precedenza, l'aggiunta di una grande quantità di memoria non può influenzare grandemente il buon funzionamento del computer: ogni componente dovrebbe essere in proporzione agli altri componenti. Si può decidere di aumentare la RAM se un'applicazione di cui si vuole fare uso richiede, come requisiti minimi una certa quantità di RAM; oppure si può optare per un'applicazione che sia meno onerosa in termini di memoria (cfr.applicazioni open-source).

7) Pensando alla struttura e al funzionamento di un calcolatore, su quali parametri si può agire per avere sistemi di elaborazione più veloci?

Sono molteplici i fattori che influenzano la velocità di elaborazione di un calcolatore.
In prima, pensando al processo di fabbricazione dei calcolatori un particolare sguardo merita la tecnologia dei materiali utilizzati, la miniaturizzazione dei componenti, i tipi di processi adottati per la produzione in serie, ecc…
Limitandosi alla struttura e il funzionamento del calcolatore, si può intervenire nelle due fasi che caratterizzano il modo di funzionare della macchina:

- Fase di *fetch*: ridurre il numero di fasi (questo comporta disporre di BUS "larghi", in modo da poter leggere l'intera istruzione in un minor numero di cicli); disporre di memorie che rispondono rapidamente (memoria principale e memoria *cache*);
- Fase di execute: avere circuiti che eseguono le operazioni rapidamente (ad esempio, coprocessore matematico per le operazioni in floating point, e altri circuiti ad hoc per altre applicazioni)

Si può intervenire complessivamente nelle due fasi con la tecnica del *pipeline*: la fase di *execute* dell'istruzione precedente e la fase di *fetch* dell'istruzione successiva *avvengono nello stesso tempo*, in parallelo.
Sistemi d'elaborazione molto veloci (rispetto ai PC standard) sono le workstation, i sistemi multiprocessore, i sistemi che utilizzano periferici e porte di comunicazione SCSI, unità di memorizzazione dei dati multi-read ad alta velocità, ecc…
Esempi di calcolatori molto veloci ci giungono dal mondo dei super computer, purtroppo la principale limitazione nell'utilizzo di calcolatori molto veloci è un dispendio economico non indifferente.

8) Cosa si intende per memoria "segmentata"? Quanto è grande solitamente un segmento?

Nei primi processori della famiglia INTEL '86 (8086, 80186, 80286) l'indirizzo è costituito da due parti, detti rispettivamente *segment* (16 bit) e *offset* (16 bit). Quando il microprocessore accede alla memoria le due parti vengono sommate (dopo aver fatto scorrere la parte di *segment* a sinistra di 4 posizioni con inserimento di zeri a destra). Anche il Program Counter è formato di queste due parti. Quando avviene l'autoincremento, l'offset aumenta di 1, ma non si ha il travaso di carry tra offset e segment. L'effetto è che il PC scandisce solo 2^{16} (= 64K) celle contigue, per poi ricominciare d'ac-

capo. Si identificano così blocchi di 64K, detti segmenti. La memoria risulta pertanto "segmentata". La motivazione di questa scelta architetturale era legata alla possibilità di implementare in hardware le protezioni richieste dai moderni sistemi d'elaborazione. Ad esempio un segmento, quello contenente il codice di un programmi, poteva essere protetto dalle scritture (protezione read-only), il segmento di stack poteva avere altre protezioni, e così via.
Lo stesso meccanismo di mancanza di travaso del carry si ha non soltanto nell'autoincremento, ma in qualsiasi calcolo di un indirizzo (ad esempio quello che, a partire dalla posizione iniziale, permette di accedere ad un elemento di un vettore). È per questo che nel TURBO-C c'è un vincolo sulla lunghezza di un vettore (e, in generale, di qualsiasi dato strutturato), il quale non può occupare più di 64K.

9) Si consideri una CPU che impieghi 4 cicli macchina per eseguire ogni istruzione ed una frequenza di clock di 400 MHz. La CPU esegue un programma che deve compiere un accesso in memoria ogni 5 istruzioni. Ipotizzando un tempo di accesso alla memoria di 60 ns, quante istruzioni verranno eseguite in 1 secondo?

Ogni gruppo di 5 istruzioni comprende un accesso alla memoria. Il tempo necessario ad eseguire 5 istruzioni, seguite da un accesso in memoria è pari a:
$T_{5i+m} = 5\ T_i + T_m$
dove:
- T_i = tempo necessario ad eseguire un'istruzione = $4 / (400 * 10^6) = 0.01\ \mu s = 10$ ns
- T_m = tempo necessario a compiere un accesso in memoria = 60 ns

Il tempo necessario a eseguire un gruppo di 5 istruzioni è pari a:
$T_{5i+m} = 5 \times 10$ ns + 60 ns = 110 ns.
In ogni secondo, vengono eseguite un numero di gruppi di 5 istruzioni pari a:
$1\ s / 110\ ns \approx 1/100 * 10^9 = 10^7$ (= 9.090.909)
Il numero di istruzioni eseguite è pari a:
$5 \times 10^7 = 50 * 10^6$ istruzioni = 50 MIPS.