1. Kod konvencije pozivanja cacdl na arhitekturi X86 potprogram tipicno pocinje sljedecim instrukcijama:
- push ebp, mov ebp,esp

2. U nekom podatkovnom registru D3 nalazi se FFFFFFFF. Što će se dogoditi ako se izvrsi sljedeca naredba : MOVE #0, D3
FFFF0000

3. Na adresi $3000 nalazi se podatak ABCDEFGH. A u heks. formi iznosi 41.
U registru A0 nalazi se adresa 3000.
Što ce se dogoditi ako se izvede sljedeci odsjecak:
ADD #2,A0
MOVE.W (A0), D
A0=00003002, D0=00004344

4. Koji slijed naredaba se mora izvesti da bi se izvela a*b+c preko stoga.
mov eax,[epb +8], imul eax,[epb +12], add eax,[epb+16]

5. Kazalo stoga na MC 68000 je:
registar A7

6. Tko prazni stog: (ili tako nesto, uglavnom bilo je ponudjeno, glavni program, potprogram na ulazu, potprogram na izlazu i slicno)
potprogram na izlazu

7. Registar stanja na MC68000:
SR - 16 bitni registar - korisnicki i sistemski

8. Sub exp,X. O cemu ovisi X:
o broju bitova lokalnih varijabli potprograma

9. Kako moraju biti pisane naredbe u EASY 68K:
moraju biti uvucene za razmak ili tabulator

10. Koliko naredbi je potrebno da bi se zbrojila 3 broja na stogu:
3

- kako se zavrsava onaj cdcel epilog (pop ebp, ret)
- ako je u D1 nesto sta kad se uradi MOVE.L #0, D1

- koji se registar koristi kao kazalo stoga
- koliko bitni je SR registar
- bio je onaj dio koda SUB #1, D1 BNE $1000, ja sam stavio FFFF0000..
- ako treba uraditi a*b+c ili tako nesto kako ide kod.... mov eax, [ebp + 8]itd...

1. Koji je standardni nacin pristupanja varijablama i parametrima :) (ebp, skoro ista recenica u pripremi)

2. u sun sparcu, koji na koji se nacin adresira najznacajniji bit 32 bitne varijable
a)*((char*)*p) (mozda je jedna zvjezdica viska al nije bitno, nije bilo slicnog rjesenje
b) *((char*)*p)+2)
c) *((long*)*p)
d)*((char*)*p)+ il minus neki veci broj, mislim 12)
sparc je big endian, sto znaci da ce najznacajniji bajt biti na pocetnoj adresi, stoga: *((char*)*p)

3. Kako bi strojno realizirali a*b+c. U pripremi je realizirano (a+b)*c, tako da je to prakticki tamo rjeseno, samo treba pazit sto je a, sto je b, sto c i gdje je stavljen imul.
a) mov eax, [ebp+12], imul eax, [ebp+8], add eax, [ebp+16] to bi mislim bilo tocno od ponudjenih ako se sjecam (neznam da li je 12 bilo prvo il 8).

a - ebp + 8
b - ebp + 12
c - ebp + 16
mov eax, [ebp+12], - stavljas b u eax
imul eax, [ebp+8], - mnozis a s eax (tj. b)
add eax, [ebp+16] - zbrajas eax s c (tj. umnozak a*b)

Iako, u ovom zadatku mi jedan dio nije bio jasan. U wikipediji pise da se stavlja na stog s desne strane, i u njihovome primjeru, kad je lista podataka bila prog(a,b,c), prvi podatak na stogu je bio c, pa b, pa a. Ali takav odgovor nije bio ponudjen medju tocnima pa onda to nisam ni gledao u zadatku, al svejedno mi je bilo cudno

4. Kako clock() prikazuje vrijeme
a) 1:10
b) 1:100
c) 1:1000
d) clock_stepnesto:1
c. jer rezolucija clock()-a je u milisekundama
c) 1:1000

5. Nesto s preskocima iz drugog zadatka. To nisam znao. Ovo po sjecanju opet.
Ako imamo veliko polje 16 bitnih (il kbitnih?!?) podataka, i dohvacamo svako 4ti u memoriju s 32kb (vjerojatno L1 onda, al to ne pise), koliko mozemo ocekivati promasaja. I sad su tu dani omjeri

a) 1:1
b) 1:2
c) 2:1
d) 3:1
tu bi trebalo znati kolika je duljina linije, jer o tome ovisi broj promasaja. svaki 4-ti 16bit podatak znaci da cita svaki osmi bajt. ako je linija 8bajtna, znaci da je svaki read promasaj. ako je 16, onda mislim da je 1:1 i tako dalje.
ovo je IMHO.

6. Kolika je "uobicajena sirina L1 medjuspremnika". Ovo pitanje mi nikako nije jasno sto su oni pod tim mislili. U onim specifikacijama sparca koji linkaju, pise
L1 Cache :
* 64 KB 4-way data
* 32 KB 4-way instruction
* 2 KB Write, 2 KB Prefetch
Al to nije to. U svakom slucaju, ponudjeni odgovori
a) 128 bit
b) 8 bit
c) 512 bit
d) 32 bit
Ostali su bili sablonski koje mozete dirkt prepisat iz pripreme ako vam dodju, nesto ovako tipa
mislim da je 32bit

7. Kako uobicajeno zavrsavamo strukturu strojnog programa, i onda neke ponudjene stvari, al to je onaj dio u prvom primjeru gdje pise cdecl epilog

Način pristupanja parametrima i lokalnim varijablama. Mislim da je odgovor **regidtar EBP**

1. Koliko je uobicajno velik l1 cache?
stavio 16kb (32 nije bilo ponudjeno)

2. Kako napisati a*b+c?

3. Promašaj i pogoci.

4. Koliko treba instrukcija da se zbroje 3 broja sa stoga?

5. Kako se vraća potprogam prema cdcel?

6. clock - kako se odnosi prema sekundi?

7. U **x86**, koji na koji se nacin adresira najznacajniji bit 32 bitne varijable

a. *((char*)*p) (mozda je jedna zvjezdica viska al nije bitno, nije bilo slicnog rjesenje
b. *((char*)*p)+2)
c. *((long*)*p)
d. *((char*)*p)+3 -> vjerojatno točno
8. Lokalne varijable se nalaze u odnosu na argumente gdje?
Vjerojatan odgovor: lokalne varijable su "ispod" ebp

kolko najmanje instrukcija za zbrojiti 3 varijable u istoj liniji memorije...

4. Koliko treba instrukcija da se zbroje 3 broja sa stoga?

7. U x86, koji na koji se nacin adresira najznacajniji bit 32 bitne varijable
a. *((char*)*p) (mozda je jedna zvjezdica viska al nije bitno, nije bilo slicnog rjesenje
b. *((char*)*p)+2)
c. *((long*)*p)
d. *((char*)*p)+3 -> vjerojatno točno

za x86 je tocno d) ako se ne varam jer je to little endian, a za sparc je tocno a) jer je on big endian

----> kolko najmanje instrukcija za zbrojiti 3 varijable u istoj liniji memorije...

---->kolko linija ima L1...??

---->kako staviti 1234 u nešto...??
mov eax, 1234

---->kako završava potprogram??
Mislim da je to kao ono moje pitanje, samo varijacija.
// cdecl epilog:
pop ebp // vrati stari ebp
ret // povratak iz potprograma

1. Koliko je uobicajno velik l1 cache? (16kb ili 32kb) ????

3. Promašaj i pogoci.

4. Koliko treba instrukcija da se zbroje 3 broja sa stoga?
Mislim 3. Stavis prvi broj s stoga u eax, zbrojis eax i drugi broj sa stoga, zbrojis eax i treci broj sa stoga. U eax je zbroj. 3. ako moze krace, e neznam.