OHJ-4200 Laitteistoläheinen ohjelmointi
Tentti 19.11.2007

1. Kerro sparc-arkkitehtuurista seuraavaa:
   1. Millainen on sparcin aktivaatiotietue?
   2. Miten tila aktivaatiotietueelle varataan?
   3. Entä vapautetaan?
   4. Aliohjelmakutsussa käytetään *branch-and-link* konekäskyä synteettisen käskyn *call* kautta. Tämä tallettaa paluuosoitteen rekisteriin *%o7*. Miksi näin kutsutusta aliohjelmasta voidaan kutsua uusia aliohjelmia ilman ongelmia?
   5. Mitä tarkoitetaan leaf-proceduurilla?
   6. Käskykanta sisältää myös käskyn *sethi*. Mihin sitä tarvitaan?

   Katso kaavioita paperin kääntöpuolelta!

2. Missä tilanteissa paikallista muuttujaa (tai parametria) ei voida allokoida rekisteriin (oletetaan C-kieli käytössä)?

3. Nykyaikaisissa käyttöjärjestelmissä dynaamiset kirjastot (tai yleisemmin *shared objects*) ovat laajalti käytössä. Miksi tälläiset kirjastot ovat
   1. paikasta riippumatonta koodia?
   2. Kirjoitussuojattua koodia?

4. Koodialueet ovat muutenkin yleisesti aina kirjoitussuojattuja, joten niitä ei enää ajoaikana voi muuttaa. Millaisella mekanismilla dynaamisen kirjaston ajonaikainen liittäminen on siis mahdollista, vaatiihan se aliohjelmakutsujen muuttamista viittamaan oikeaan paikkaan?

5. Ohjelmaa Unix (Linux) ympäristössä luotaessa käännösprosessi tuottaa lähdekoodista suoritettavan tiedoston.
   1. Mitkä vaiheet prosessiin kuuluvat?
   2. Missä vaiheessa sijainti muistiavaruudessa lyödään lukkoon?
      1. Sovelluskoodille?
      2. Staattisesti ladatulle kirjastolle?
      3. Dynaamisesti ladatulle kirjastolle
      4. Data-alueille?

in	%i7	(%r31)	return address – 8 †
	%fp, %i6	(%r30)	frame pointer †
	%i5	(%r29)	incoming param 6 †
	%i4	(%r28)	incoming param 5 †
	%i3	(%r27)	incoming param 4 †
	%i2	(%r26)	incoming param 3 †
	%i1	(%r25)	incoming param 2 †
	%i0	(%r24)	incoming param 1 / return value to caller †
local	%l7	(%r23)	local 7 †
	%l6	(%r22)	local 6 †
	%l5	(%r21)	local 5 †
	%l4	(%r20)	local 4 †
	%l3	(%r19)	local 3 †
	%l2	(%r18)	local 2 †
	%l1	(%r17)	local 1 †
	%l0	(%r16)	local 0 †
out	%o7	(%r15)	temporary value / address of CALL instruction ‡
	%sp, %o6	(%r14)	stack pointer †
	%o5	(%r13)	outgoing param 6 ‡
	%o4	(%r12)	outgoing param 5 ‡
	%o3	(%r11)	outgoing param 4 ‡
	%o2	(%r10)	outgoing param 3 ‡
	%o1	(%r9)	outgoing param 2 ‡
	%o0	(%r8)	outgoing param 1 / return value from callee ‡
global	%g7	(%r7)	global 7 (SPARC ABI: use reserved)
	%g6	(%r6)	global 6 (SPARC ABI: use reserved)
	%g5	(%r5)	global 5 (SPARC ABI: use reserved)
	%g4	(%r4)	global 4 (SPARC ABI: global register variable §)
	%g3	(%r3)	global 3 (SPARC ABI: global register variable §)
	%g2	(%r2)	global 2 (SPARC ABI: global register variable §)
	%g1	(%r1)	temporary value ‡
	%g0	(%r0)	0
state	%y		Y register (used in multiplication/division) ‡
	(*icc* field of %psr)		Integer condition codes ‡
	(*fcc* field of %fsr)		Floating-point condition codes ‡
	(*ccc* field of %csr)		Coprocessor condition codes ‡
floating point	%f31		floating-point value ‡
	:		:
	:		:
	%f0		floating-point value ‡

† assumed by caller to be preserved across a procedure call

‡ assumed by caller to be destroyed (volatile) across a procedure call

§ should not be used in SPARC ABI library code

Figure D-1 *SPARC Register Set, as Seen by a User-Mode Procedure*

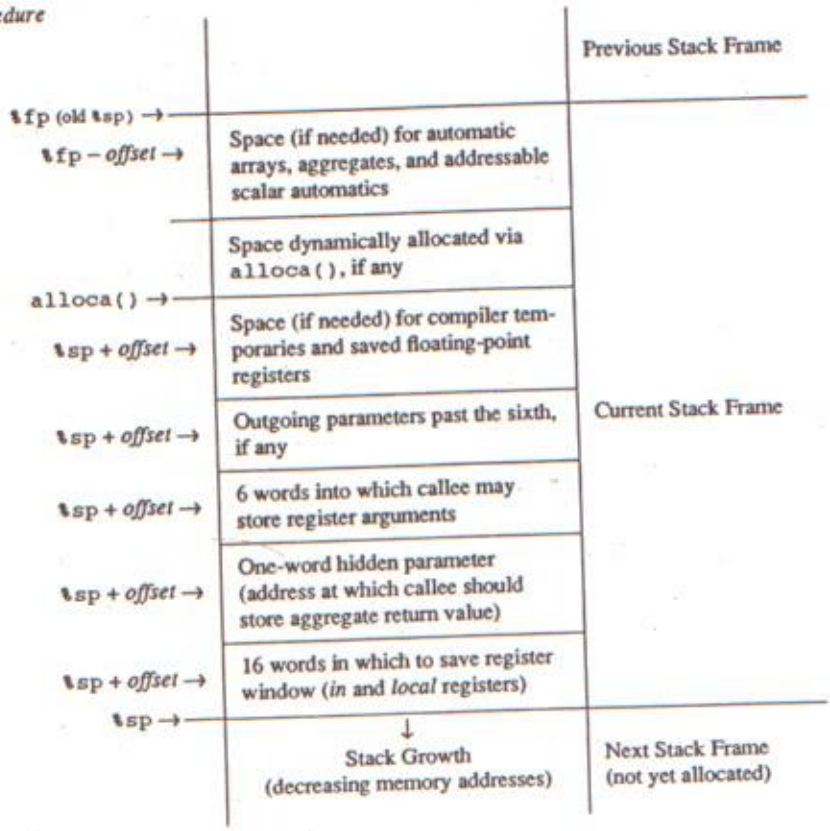

Figure D-2 *The User Stack Frame*