STRUTTURE PROGRAMMA E DIRETTIVE ASSEMBLATORE

→ definiamo il modello di **architettura run-time**:
- destinazione di uso dei registri;
- colloc e ingombro diverse classi di var.;

→ programma formato da: codice (main, funz. ut.), dati (var glob e dim), pila (area di attivaz.);

→ **DICHIARARE I SEGMENTI**: .data: segmento dati
.text: funz e var glob
main: cod. programma

→ nel file sorgente (var e strutt d'alloc., text seg, indic. simb glob ecc.) sono specificate le direttive dell'assl per generare il file ogg.

→ **DICHIARARE VARIABILI**: — var. globale nominale, tramite indr simbolico (=suo nome) viene collocata in mem da linkr o assbl
es. int a=1 → A: .word 1 in '.data'
char c='@' → C: .byte 64

•SYSTEM-CALL:
↳ formata da: codice, argomenti (opz.), valori di ritorno;
→ per richiederne un **servizio**:
- carico codice syscall in $v0;
- carico arg. in reg. $a0-$a3;
- esegue syscall;
- carico (se nec.) valore di ritorno in $v0;

SOTTOPROGRAMMI E MIPS

→ la chiamata a sottoprogrammi ha come effetto la creazione di un **record di attivazione** sullo **stack**;
→ **istr. macchina** per gestione spr:
- chiamante (caller), gestisce passaggio parametri e Chiamata sottoprogrammi;
- esecuzione istr. chiam. ISA, gestisce salvataggio PC e attiva sottoprogr;
- chiamato (callee), gestisce alloc. var. globali e valore restituito;

→ **MODELLO CHIAMATA A SOTTOPR**: varia a seconda del processore (consid. ISA in MIPS) ← vincoli da rispettare riguardo passaggio param, val restituito, indr ritorno,
i **gruppi di registri** vengono etichettati tram **callee-saved registers** per preservali al Chiamante
(il chiamato utilizza solo registri temp, arg e di ritorno specifici in modo da non intaccare il chiam.)

→ **area (frame) di attivazione**: spazio richiesto dal chiamato in byte per salvat. registri e alloc var globali;
↳ fp salvato solo se per referenziare mem in stack
↳ ra non viene salvato in procedura foglia;

→ **VAR LOCALE NOMINALE**: gestita in base al suo utilizzo, al tipo e al grado di ottimizzazione del codice;

- VAR SCALARE O PUNTATORE → in reg o frame attivazione;
- VAR SCALARE E PUNT → frame attivazione;
- VAR ARRAY O STRUCT → frame attivazione (per array calloc. su stack con a[0] in basso)

PROLOGO CHIAMANTE → scrive in a0-a3 param in ingr
↓
se nec. salva t0-t9 sullo stack
↓
se nec anche a0-a3 vengono prec. salvati
↓
jal funz

PROLOGO CHIAMATO → crea area attiva (addiu $sp, $sp, -dimByte)
↓
se $fp in uso viene salvato (su pila) e aggiornato a nuova cima
↓
se f non è foglia, $ra viene salvato su pila (→ quindi c'è chiamata annidata)
↓
salva su pila reg s0-s7 per var locali
↓
CORPO ELAB. CHIAMATO
↓
EPILOGO CHIAMATO
↓
scrive in v0 ritorno
↓
ripristina s0-s7 per var locali e $fp e $ra(se nec)
↓
elimina frame → addiu $sp, $sp, dim byte
↓
rientro al chiamante → jr $ra

↳ **EPILOGO CHIAMANTE**: ripristina a0-a3 e t0-t9 e trova v0 risultato funz