Tema 04: Arquitectura del Set de Instrucciones (ISA)

Arquitectura de Computadoras

Ing. Nicolás Majorel Padilla (npadilla@herrera.unt.edu.ar)

http://microprocesadores.unt.edu.ar/arqcom/

Temas que veremos

- Evolución histórica de ISAs.
- ¿Qué es un ISA?
  - Formatos de Instrucciones
  - Clases de instrucciones
  - Modos de direccionamiento
  - Lenguajes de Alto Nivel
- Distintos tipos de ISA
- Pasaje de parámetros entre programas
  - Manejo del Stack y de Subrutinas
  - ABI y convención de llamadas

Lectura recomendada

- Computer Organization and Design, RISC-V Edition (2da ed, 2021)
  - Sección 2.1: *Introduction*
  - Sección 2.24: *Historical Perspective and Further Reading*

Evolución Histórica de ISAs

- Prehistoria (1960): instrucciones simples.
- 1970-1980's: era CISC.
  - Instrucciones cada vez más poderosas.
  - x86: Intel y todos sus clones.
- 1990's-actualidad: era RISC.
  - Vuelta a lo simple, motivado por **Performance**.
    - MIPS, SPARC, PowerPC, ARM, DEC Alpha, etc.
  - Conviven con CISC, que *mutan para sobrevivir*.
    - Mantienen compatibilidad, pero internamente ejecutan como RISC.

Evolución Histórica de ISAs

- En la actualidad, hay **dos ISAs dominantes**:
  - **x86_64**, en PCs, Servidores y Supercomputadoras.
    - 99% del mercado (no solamente Intel).
    - Diseñado originalmente en 2003 por… AMD.
  - **ARM** en sistemas embebidos, celulares, etc.
    - 99% de los chips de celulares y tablets.
    - Pero en realidad, hay varios ISAs distintos:
      - Thumb, Thumb-2, ARMv7, ARMv8, …
      - Usados en distintos tipos de sistemas embebidos.
- Estos ISAs son propietarios.
  - Lo que implica que para usarlos, hay que pagar licencias ($$$).

Evolución Histórica de ISAs

- *¿Por qué Intel no puede vender chips para celulares?*
- O, *¿por qué a ARM le cuesta tanto meterse en el negocio de las laptops/desktops/servers?*
- Porque el **ISA es la interfaz más importante en un sistema de computadoras**.
  - Es donde el software se encuentra con el hardware.
  - Es un “contrato implícito” desde hace 60 años:
    - El hardware puede cambiar radicalmente, pero el código que se ejecutaba en la máquina de ayer **tiene que ejecutarse exactamente igual en la de mañana**, pero más rápido.
    - Aunque cambie, el hardware siempre se muestra igual para el software, “hablando el mismo idioma”.
    - *“Novel hardware is great, but all the performance and features in the world won’t do you much good if commonly used software doesn’t run on it.”*

Evolución Histórica de ISAs

- Hay dos nuevas tendencias en los últimos años.
- ARM está ingresando en el otro mercado
  - De la mano de los procesadores M de Apple (Nov 21).
  - De la mano de Fugaku, actual n° 4 en el ranking Top500 (Nov 23), que fue n° 1 en 2021.
- Hay un tercer ISA ganando mercado: **RISC-V**.
  - Originado hace casi 15 años en una Universidad.
  - Se está acercando rápidamente a los valores de performance, costo y consumo de energía de los chips de ARM.
  - Recibió grandes apoyos económicos por parte de Intel (Feb 22).
  - Lo veremos a fondo en el Tema 06.

Componentes de un ISA

- El ISA es conocido como ***el Modelo del Programador de la Máquina.***
- Es un nivel de abstracción que permite separar el desarrollo del software del desarrollo del hardware.
- Es la información que se requiere para escribir programas para un determinado procesador.

- Está compuesto por 4 componentes:

1. **Celdas de almacenamiento**
   - Registros de propósito general y especial del CPU.
   - Muchas celdas de propósito general de igual tamaño en Memoria.
   - Almacenamiento relacionado con los dispositivos de I/O.

Componentes de un ISA

2. Formatos de las instrucciones

- Tamaño y significado de los diferentes campos de las instrucciones.
- Modos de direccionamiento soportados.

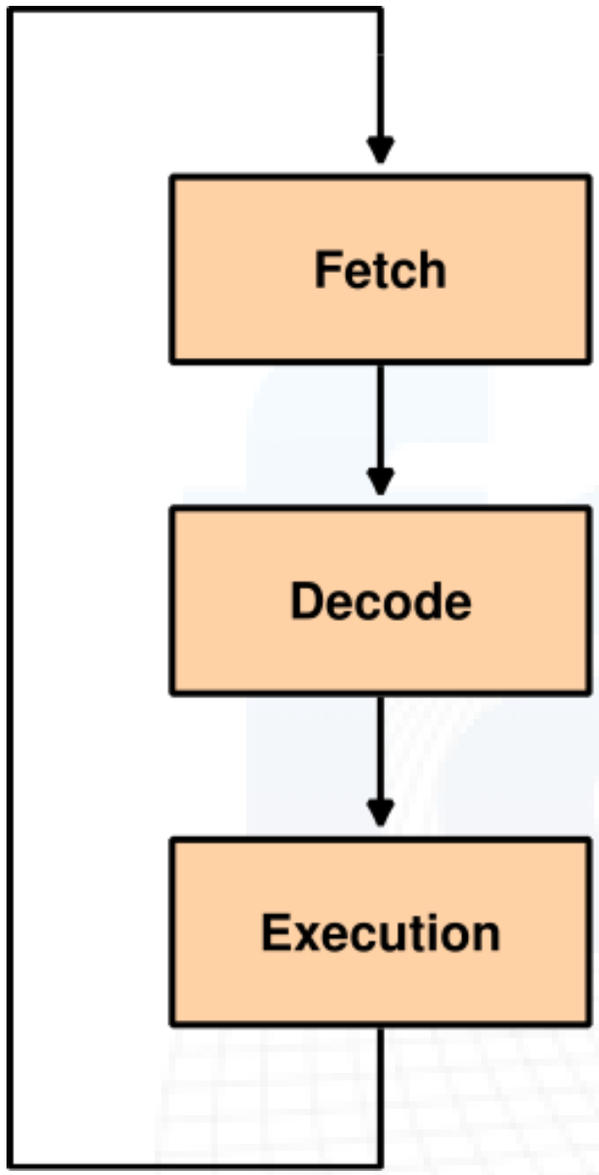
3. El conjunto (Set) de instrucciones

- Repertorio completo de las operaciones de la máquina.
- Usa celdas de almacenamiento, formatos y resultados del ciclo de la instrucción.

4. Naturaleza del ciclo de la instrucción

- Cosas que se hacen independientemente de la instrucción en cuestión.

Repaso: Ciclo de la Instrucción (SMM)

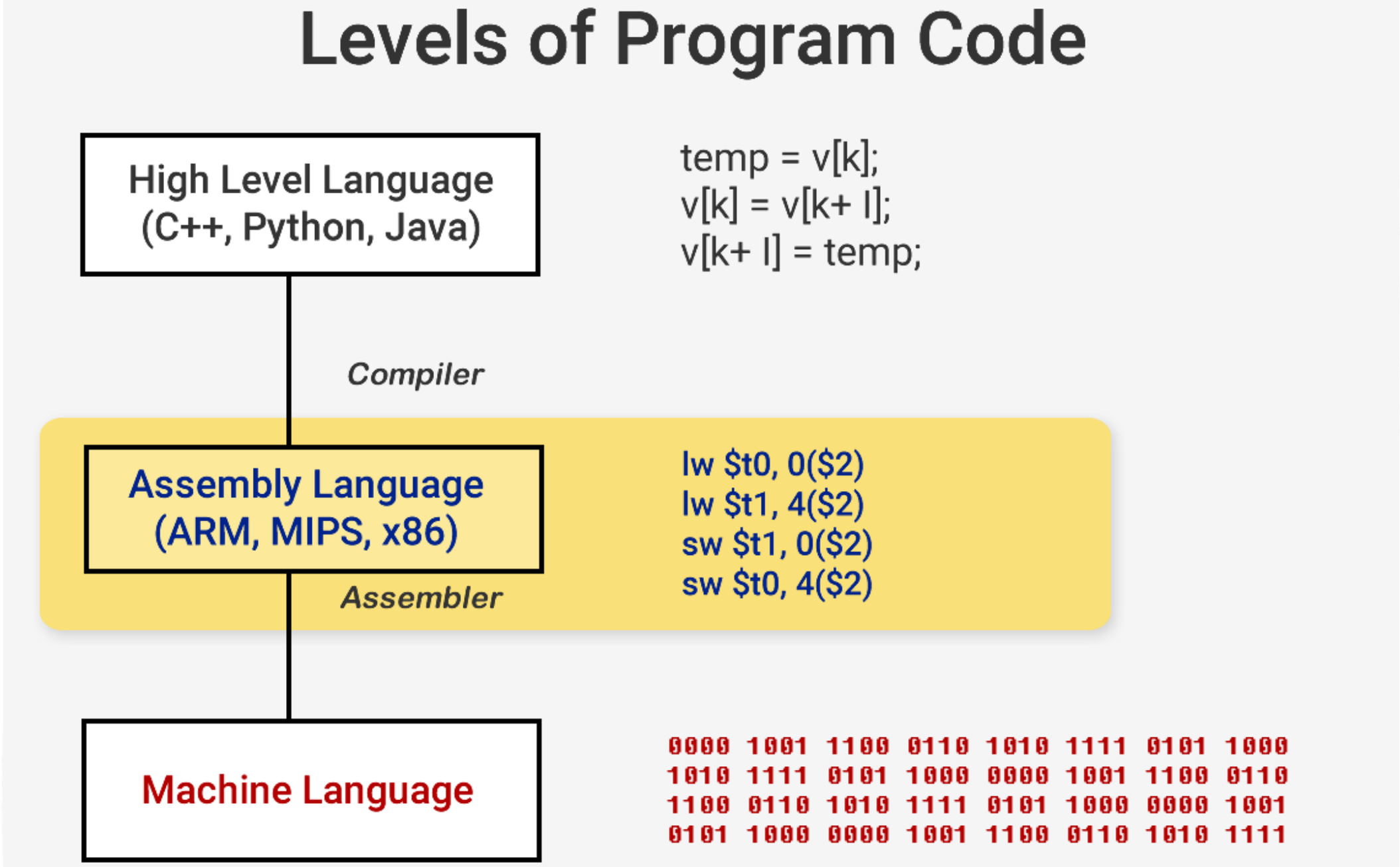


- Obtener la instruccion de memoria M[PC]
- Guardar la instruccion en el registro IR
- Actualizar el puntero a la siguiente instrucción PC

- Determinar las acciones que se requieren para que la instrucción genere el resultado esperado

- Ejecutar las acciones determinadas en el paso anterior para qobtener el resultado esperado (Leer o escribir en memoria, Procesar en la ALU, etc)

Repaso: Niveles de código



¿Qué debe especificar una instrucción?

Flujo de datos

```
ADD R2, R1, R3
```

```
ADD R2, R1, R3
```

```
ADD R2, R1, R3
```

```
ADD R2, R1, R3
BR endloop
```

- **¿Qué operación realizar?**
  - Op code: add, load, branch, etc.
- **¿Dónde están los operando/s?**
  - En registros del CPU, celdas de memoria, lugares de I/O o parte de una instrucción.
- **¿Dónde se guarda el resultado?**
  - En un registro, M, I/O o Inst.
- **¿Dónde está la próxima instrucción?**
  - Lugar de memoria al que apunta el PC
  - Para ej. salto – PC ⇐ endloop

Formato de una Instrucción (SMM)

- Dividir la instrucción de *n* bits en campos de distinta longitud.
- Op-Code: Indica qué operación se realizará.
  - Si tiene un tamaño de *b* bits, *¿cuántas operaciones distintas pueden soportarse?*
- Operandos: Con qué datos se realiza la operación, tanto fuente/s como destino/s.
  - En general desde 0 a 3 operandos.
  - El dato puede estar en otro campo de la Instrucción, en un Registro o en Memoria.
  - Indican cómo se rutea la cantidad necesaria de bits hacia y desde un componente de HW que se encarga de realizar la operación deseada.
- Puede haber más de un formato, dependiendo del Op-Code.

Clases de Instrucciones (SMM)

- **Movimiento de Datos**
  - Sirven para transferir datos desde registros o memoria, hacia registros o memoria.
  - Tienen siempre una fuente y un destino.
  - **Load:** La fuente es memoria y el destino un registro.
  - **Store:** La fuente es un registro y el destino es memoria.
  - Hay casos con fuentes y destino ambos M o ambos R.

Clases de Instrucciones (SMM)

- **Procesamiento**
  - Instrucciones Aritméticas y Lógicas.
  - Procesar uno o más operandos fuentes y guardar el resultado en un destino.
  - **add**, **sub**, **shift**, etc.
- **Control de flujo de Instrucciones (Saltos)**
  - Alterar el flujo normal de control en lugar de ejecutar la siguiente instrucción.
  - Pueden ser saltos condicionales o incondicionales.
  - **branch**, **jump**, **bnez**, etc.
- **Misceláneas**
  - Otras funciones no encasilladas en los tipos anteriores.

Modos de direccionamiento

- Son el soporte de Hw para una forma útil de determinar dónde se encuentra el operando.
- Diferentes modos de direccionamiento resuelven distintos problemas de los lenguajes de alto nivel.
  - Algunas variables se conocen en tiempo de compilación.
  - Otras se conocen recién en tiempo de ejecución (p.ej: punteros)
  - Puede ser necesario calcular las direcciones (p.ej: Componentes de una tabla o vector, o estructura o registro)
  - Es posible almacenar valores constantes en la misma instrucción, o adyacente, sin usar memoria.

Modos de direccionamiento

- Por ejemplo, en C se accede a v[i]
  - v es un puntero a la dirección base del vector, es constante.
  - i es el índice variable.
    - No puede superar el tamaño del vector.
  - Para poder acceder a v[i] se necesita una "aritmética de direcciones"*:*
    - Dirección base de v[] + i * tamaño de los elementos de v[]
- *¿Cuántos modos de direccionamiento tiene el ISA Thumb-2 de ARM?*

Clasificación de ISAs

- Las diferentes arquitecturas generalmente se clasifican en cómo y dónde se ubican los operandos, y cómo estos operandos son especificados por la instrucción.
- Algunos registros tienen como una “personalidad” propia.
  - PC, Acumulador, Stack Pointer.

- Clasificación basada en las instrucciones aritméticas que tienen dos operandos y un resultado.
- ISAs de más de 3 operandos, no se ven actualmente.
  - Permiten que la dirección de la próxima instrucción se especifique explícitamente.

Clasificación de ISAs

- Un ISA de 3 operandos usa modos de direccionamiento tanto para los operandos fuente como para el resultado.
  - Op3 ← Op1 op Op2
- Un ISA de 2 operandos usa el operando destino como uno de los operando fuente
  - Op1 ← Op1 op Op2
- Un ISA de 1 operando emplea de manera implícita un registro llamado acumulador
  - Contiene tanto un operando fuente como el resultado
  - Acc ← Acc op Op1
- Un ISA de 0 operandos emplea una pila para contener los operandos
  - ToS ← ToS op SoS

Comparación de los distintos ISAs

- 0 y 1 operandos: instrucciones más cortas.
- 2 y 3 operandos: performance.

- Posibles métricas:
  - Tamaño de las Instrucciones.
  - Cantidad de Instrucciones del ISA.
  - Tamaño del programa.
  - Cantidad de accesos a Memoria.
  - *¿Cuál es mejor?*

Pasaje de parámetros entre programas

- Se denomina procedimiento “llamante” (***caller***) a aquel que como parte de su ejecución invoca a uno o más procedimientos.
- Se denomina procedimiento “llamado” (***callee***) a aquel que es invocado por otro procedimiento.
  - Un programa puede ser *caller* y *callee*, es lo más común.
  - Los procedimientos que son solamente llamados se denominan procedimientos “hoja”.
- Se denomina ***Application Binary Interface (ABI)*** a un estándar para el uso de los registros y del mapa de memoria.
  - Define “funciones” estandarizadas para los registros.
  - Permite a los programas interactuar eficientemente aún sin ser desarrollados ni compilados de manera conjunta (interoperabilidad).

Convención de llamadas a subrutinas

- Se denomina “Convención de llamadas” al subconjunto de la ABI que especifica cómo se pasan datos de un programa a otro.
  - Algunos registros deben ser preservados entre las llamadas a subrutinas, y otros no.
  - Si un programa o subrutina *callee* planea utilizar un registro que debe ser preservado, debe guardar en algún lugar de memoria su valor al inicio, y restablecer este valor antes de retornar.
  - Si un programa o subrutina *caller* planea utilizar un registro que no debe ser preservado entre llamadas, debe ser consciente que otro programa podría modificar su valor.

Convención de llamadas a subrutinas

- Seis pasos generales:
  1. El programa *caller* debe poner los argumentos en algún lugar donde la subrutina *callee* pueda accederlos.
  2. El programa *caller* invoca a la subrutina *callee*.
  3. La subrutina *callee* debe reservar el espacio de memoria requerido y almacenar los registros necesarios.
  4. La subrutina *callee* debe realizar su tarea requerida.
  5. La subrutina *callee* debe poner su resultado en algún lugar accesible por el programa *caller*, restaurar los registros guardados y liberar la memoria reservada.
  6. La subrutina *callee* debe retornar el control al punto de origen del programa *caller*.
- El paso 3 se conoce como prólogo de la subrutina.
- Los pasos 5 y 6 se conocen como epílogo de la subrutina.

Manejo de la pila (stack)

- Es el método estándar para pasaje de parámetros entre distintos programas.
  - Además de usar los registros, obviamente.
- Usado para guardar el estado de una tarea en caso de excepciones o interrupciones.
- El stack se implementa en memoria.
- Un ISA puede manejar el stack de manera implícita o explícita.
- Los ISA con Stack explícito tienen:
  - Un registro especial en el CPU: Stack Pointer (SP).
  - Instrucciones dedicadas: PUSH y POP.
- Los ISA sin Stack explícito lo implementan mediante:
  - Un registro de propósito general (de uso reservado)
  - Incrementos y decrementos de punteros.

Manejo de Subrutinas

- Las arquitecturas con Stack explícito poseen instrucciones específicas:
  - CALL
    - Guarda el PC en el Stack. Algunas también guardan CC.
    - Guarda el estado actual.
    - Los argumentos los guarda quien llama.
  - RETURN
    - Primero quien es llamado debe restaurar lo que guardó.
    - Luego esta instrucción restaura PC.
- Los ISA sin Stack explícito igual tienen que manejar interrupciones y subrutinas.
  - Una instrucción JAL r, que guarda el PC en el registro r.
  - Tienen otra instrucción J r, que sirve para retornar.

Resumen final

- **ISA es la interfaz más importante de todo sistema de computadoras.**
- **ISA = celdas de memoria + formatos de instrucciones + conjunto completo de instrucciones + particularidades**
- Toda instrucción debe especificar: qué operación realiza, dónde están los operandos, dónde se guarda el resultado, y dónde está la próxima instrucción.
- Todas las máquinas tienen **4 clases de instrucciones**.
- Existen ISA de 3, 2, 1 y 0 operandos.
  - Las máquinas pueden ser con registros de propósito general (GPR) o con registros con “personalidad”.

Resumen final

- **Modos de direccionamiento** para manejar en forma eficiente las estructuras de datos.
  - Requieren una aritmética de direcciones.
  - Hay diversos en la actualidad, con diferente sintaxis y semántica.
- Los Stacks se usan para manejo de subrutinas y excepciones.
  - Pueden ser explícitos o implícitos.
  - Es un método standard para pasaje de parámetros.

Agradecimientos

- Las diapositivas de este tema fueron basadas en las realizadas por el Ing. Daniel Cohen.
- A su vez inspiradas en las clases del curso CS152 de la Universidad de Berkeley, California, USA.
  - Realizadas por los Prof. D. A. Patterson, John Lazzaro, Krste Asanovic.