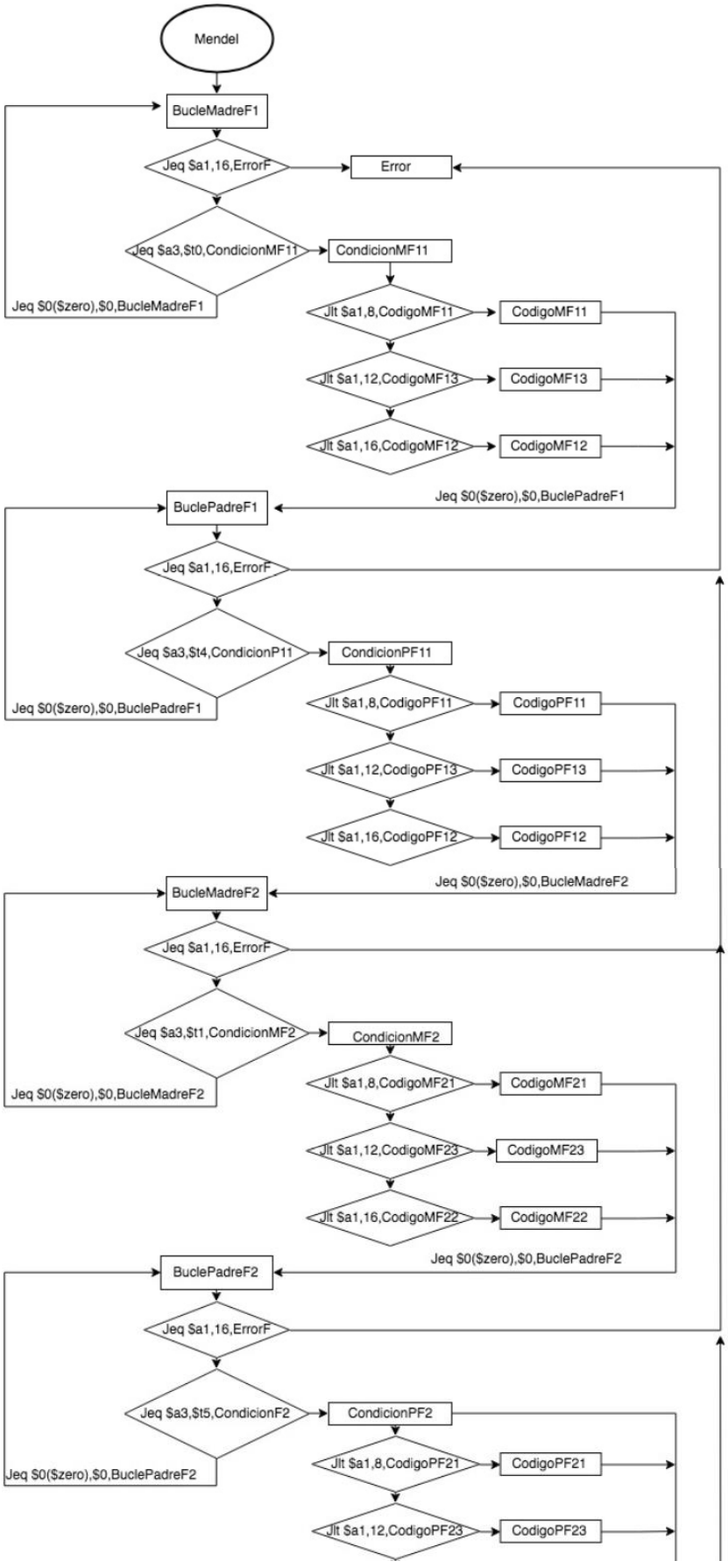
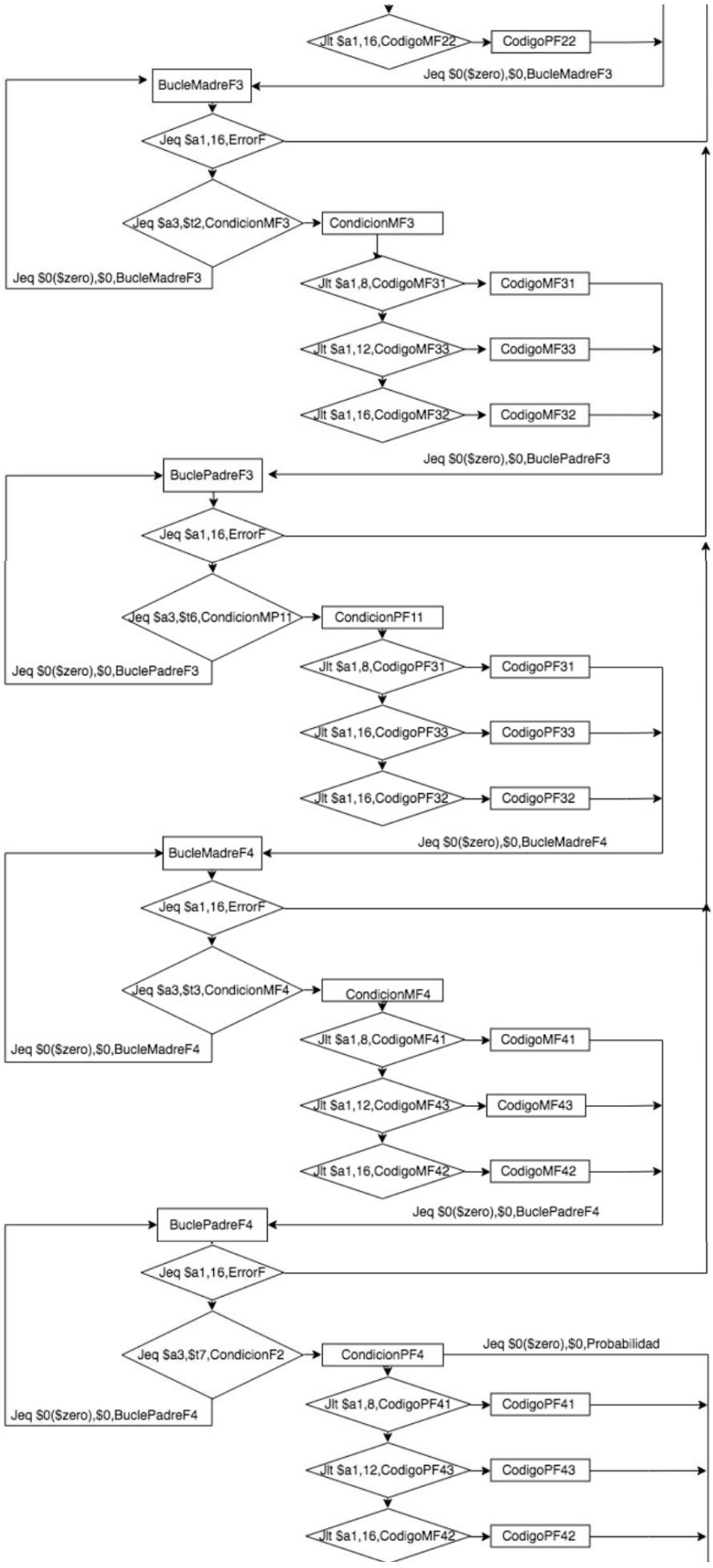
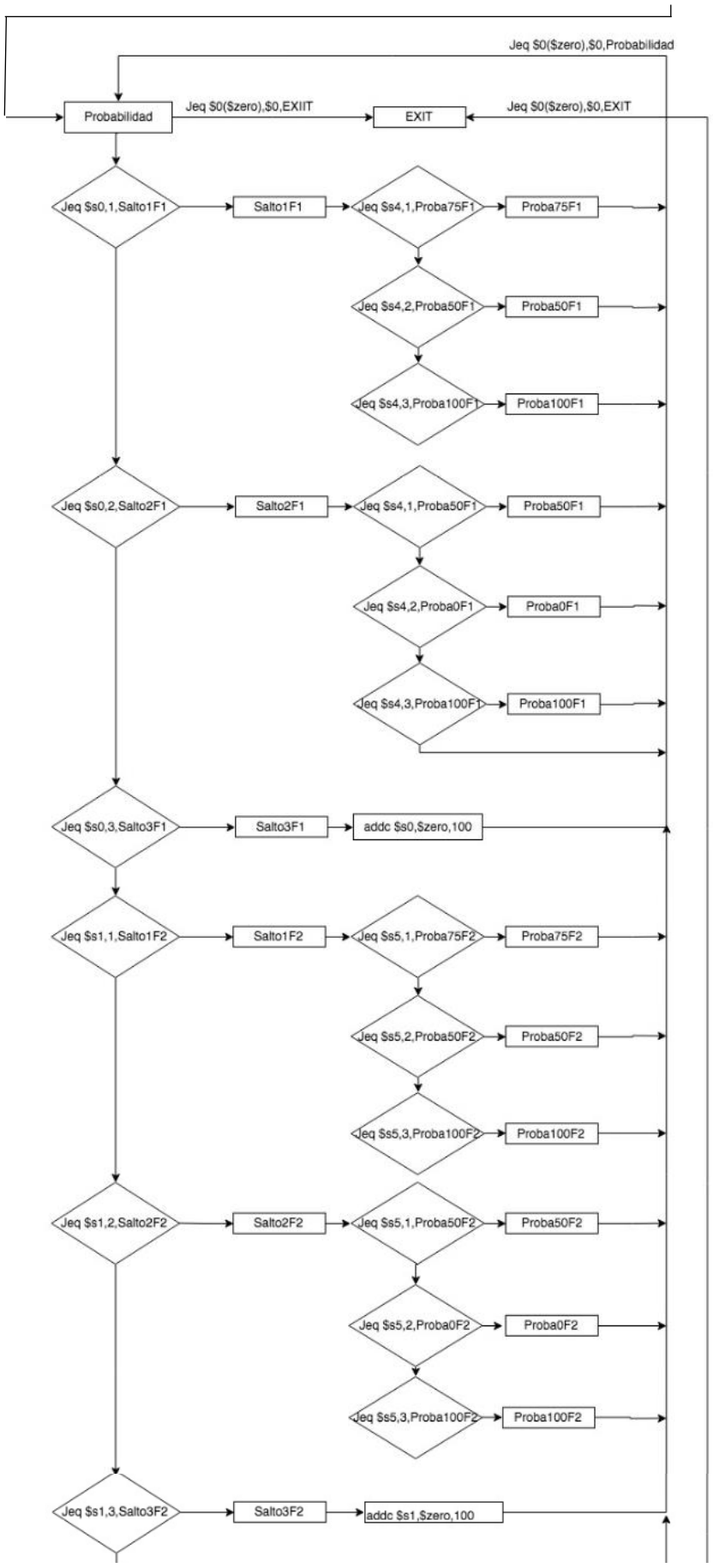
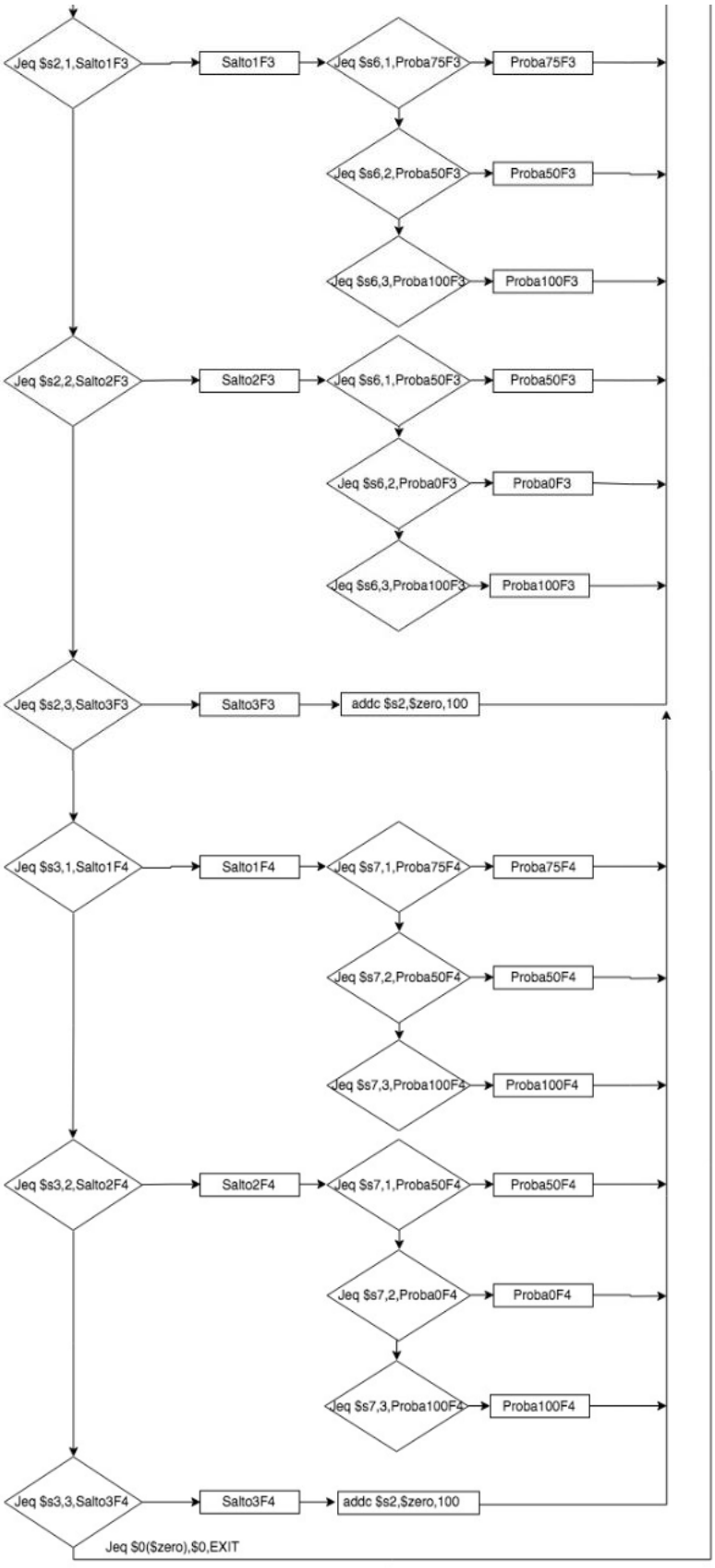
Diagrama de flujo del programa a ejecutar por el procesador




Jlt $a1,16,CodigoMF22
CodigoPF22
Jeq $0($zero),$0,BucleMadreF3
BucleMadreF3
Jeq $a1,16,ErrorF
Jeq $a3,$t2,CondicionMF3
CondicionMF3
Jeq $0($zero),$0,BucleMadreF3
Jlt $a1,8,CodigoMF31
CodigoMF31
Jlt $a1,12,CodigoMF33
CodigoMF33
Jlt $a1,16,CodigoMF32
CodigoMF32
Jeq $0($zero),$0,BuclePadreF3
BuclePadreF3
Jeq $a1,16,ErrorF
Jeq $a3,$t6,CondicionMP11
CondicionPF11
Jeq $0($zero),$0,BuclePadreF3
Jlt $a1,8,CodigoPF31
CodigoPF31
Jlt $a1,16,CodigoPF33
CodigoPF33
Jlt $a1,16,CodigoPF32
CodigoPF32
Jeq $0($zero),$0,BucleMadreF4
BucleMadreF4
Jeq $a1,16,ErrorF
Jeq $a3,$t3,CondicionMF4
CondicionMF4
Jeq $0($zero),$0,BucleMadreF4
Jlt $a1,8,CodigoMF41
CodigoMF41
Jlt $a1,12,CodigoMF43
CodigoMF43
Jlt $a1,16,CodigoMF42
CodigoMF42
Jeq $0($zero),$0,BuclePadreF4
BuclePadreF4
Jeq $a1,16,ErrorF
Jeq $a3,$t7,CondicionF2
CondicionPF4
Jeq $0($zero),$0,Probabilidad
Jeq $0($zero),$0,BuclePadreF4
Jlt $a1,8,CodigoPF41
CodigoPF41
Jlt $a1,12,CodigoPF43
CodigoPF43
Jlt $a1,16,CodigoMF42
CodigoPF42


Jeq $0($zero),$0,Probabilidad
Probabilidad
Jeq $0($zero),$0,EXIIT
EXIT
Jeq $0($zero),$0,EXIT
Jeq $s0,1,Salto1F1
Salto1F1
Jeq $s4,1,Proba75F1
Proba75F1
Jeq $s4,2,Proba50F1
Proba50F1
Jeq $s4,3,Proba100F1
Proba100F1
Jeq $s0,2,Salto2F1
Salto2F1
Jeq $s4,1,Proba50F1
Proba50F1
Jeq $s4,2,Proba0F1
Proba0F1
Jeq $s4,3,Proba100F1
Proba100F1
Jeq $s0,3,Salto3F1
Salto3F1
addc $s0,$zero,100
Jeq $s1,1,Salto1F2
Salto1F2
Jeq $s5,1,Proba75F2
Proba75F2
Jeq $s5,2,Proba50F2
Proba50F2
Jeq $s5,3,Proba100F2
Proba100F2
Jeq $s1,2,Salto2F2
Salto2F2
Jeq $s5,1,Proba50F2
Proba50F2
Jeq $s5,2,Proba0F2
Proba0F2
Jeq $s5,3,Proba100F2
Proba100F2
Jeq $s1,3,Salto3F2
Salto3F2
addc $s1,$zero,100


Jeq $s2,1,Salto1F3
Salto1F3
Jeq $s6,1,Proba75F3
Proba75F3
Jeq $s6,2,Proba50F3
Proba50F3
Jeq $s6,3,Proba100F3
Proba100F3
Jeq $s2,2,Salto2F3
Salto2F3
Jeq $s6,1,Proba50F3
Proba50F3
Jeq $s6,2,Proba0F3
Proba0F3
Jeq $s6,3,Proba100F3
Proba100F3
Jeq $s2,3,Salto3F3
Salto3F3
addc $s2,$zero,100
Jeq $s3,1,Salto1F4
Salto1F4
Jeq $s7,1,Proba75F4
Proba75F4
Jeq $s7,2,Proba50F4
Proba50F4
Jeq $s7,3,Proba100F4
Proba100F4
Jeq $s3,2,Salto2F4
Salto2F4
Jeq $s7,1,Proba50F4
Proba50F4
Jeq $s7,2,Proba0F4
Proba0F4
Jeq $s7,3,Proba100F4
Proba100F4
Jeq $s3,3,Salto3F4
Salto3F4
addc $s2,$zero,100
Jeq $0($zero),$0,EXIT

Criterios de diseño
Informe

Definiciones:

Gametos: Células sexuales encargadas de la reproducción, gametos femeninos (óvulos), gametos masculinos (espermatozoides)**.**

Cigoto: Célula que resulta de la unión de las células sexuales masculina y femenina y a partir de la cual se desarrolla el embrión de un ser vivo.

Fenotipo: Conjunto de caracteres visibles que un individuo presenta como resultado de la interacción entre su genotipo y el medio.

Genotipo: Conjunto de los genes que existen en el núcleo celular de cada individuo.

Dominante y recesivo: Este concepto alude al vínculo que establecen los alelos que forman parte de un mismo gen cuando uno de estos alelos consigue enmascarar la manifestación del fenotipo del otro alelo. El alelo que logra imponerse es el alelo **dominante**, mientras que el otro es el alelo **recesivo**.

Nota: Un alelo **dominante** y otro **recesivo**, para un determinado gen, tendrá el fenotipo **dominante**.

Generaciones Filiales: La Primera Generación Filial es la descendencia resultante del apareamiento controlado de la **generación parental**. Se da usualmente entre padres diferentes con genotipos relativamente puros.

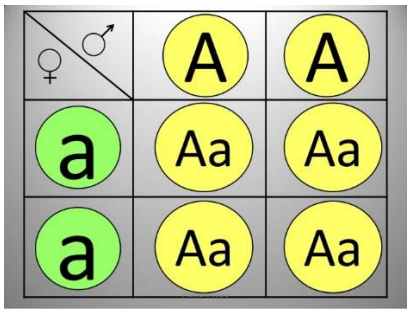
- Cuando se cruzan dos individuos provenientes de la Primera Generación Filial (F1) su descendencia se llama Segunda Generación Filial (F2)
- Cuando se cruzan dos individuos provenientes de la Segunda Generación Filial (F2) su descendencia se llama: Tercera Generación Filial (F3).

Introducción

El proyecto tiene como objetivo dar a conocer la probabilidad de que los hijos hereden ciertos rasgos físicos dominantes de los padres, de acuerdo con los cruces entre sus genotipos.

Para ello el proyecto se apoyará en dos Leyes de Mendel:

La primera establece que si se cruzan dos razas puras (un homocigoto dominante con un homocigoto recesivo) para un determinado rasgo, los descendientes de la primera generación serán todos híbridos iguales entre sí, fenotípica y genotípicamente, e iguales fenotípicamente a uno de los progenitores (de genotipo dominante), independientemente de la dirección del cruzamiento. Expresado con letras mayúsculas las dominantes (**A** = amarillo) y minúsculas las recesivas (**a** = verde), se representaría como se muestra a continuación:



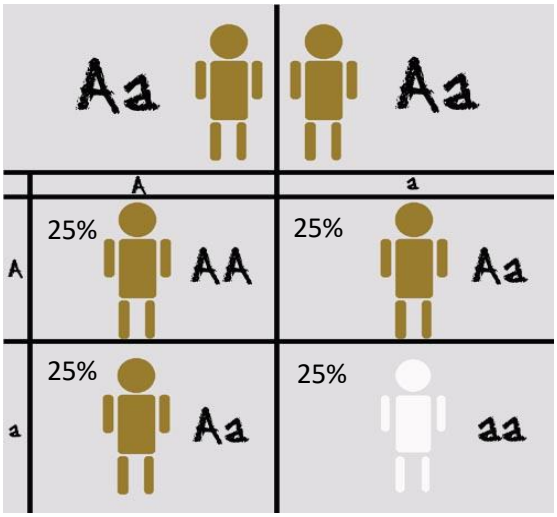
Todos los descendientes saldrían con el gen dominante.

La segunda indica que dos alelos de un mismo gen (Aa), se separan al formarse los gametos, es decir la A y la a se separan y terminan en cigotos distintos. Esto equivale a cruzar una Primera Generación Filial (Aa X Aa) con otra Primera Generación Filial (Aa X Aa), en otras palabras, cruzar individuos heterocigotos. como resultado obtuvo la Segunda Generación Filial.

EJEMPLO:

A: Cabello negro (Dominante)

a: Cabello rubio (Recesivo)



GENOTIPO	PROBABILIDAD	FENOTIPO
AA	25%	Cabello negro
Aa	50%	Cabello negro
aa	25%	Cabello rubio

El 75% saldría con el gen dominante y el 25% con el gen recesivo

Criterios de Diseño

El diseño de los datos:

El tamaño de los datos es de 9 bits. Se decidió que el usuario ingrese los genotipos a través de switches y un botoón de control (Enter) donde si es dominante pone el Switch en 1 y si es recesivo pone el Switch en 0.

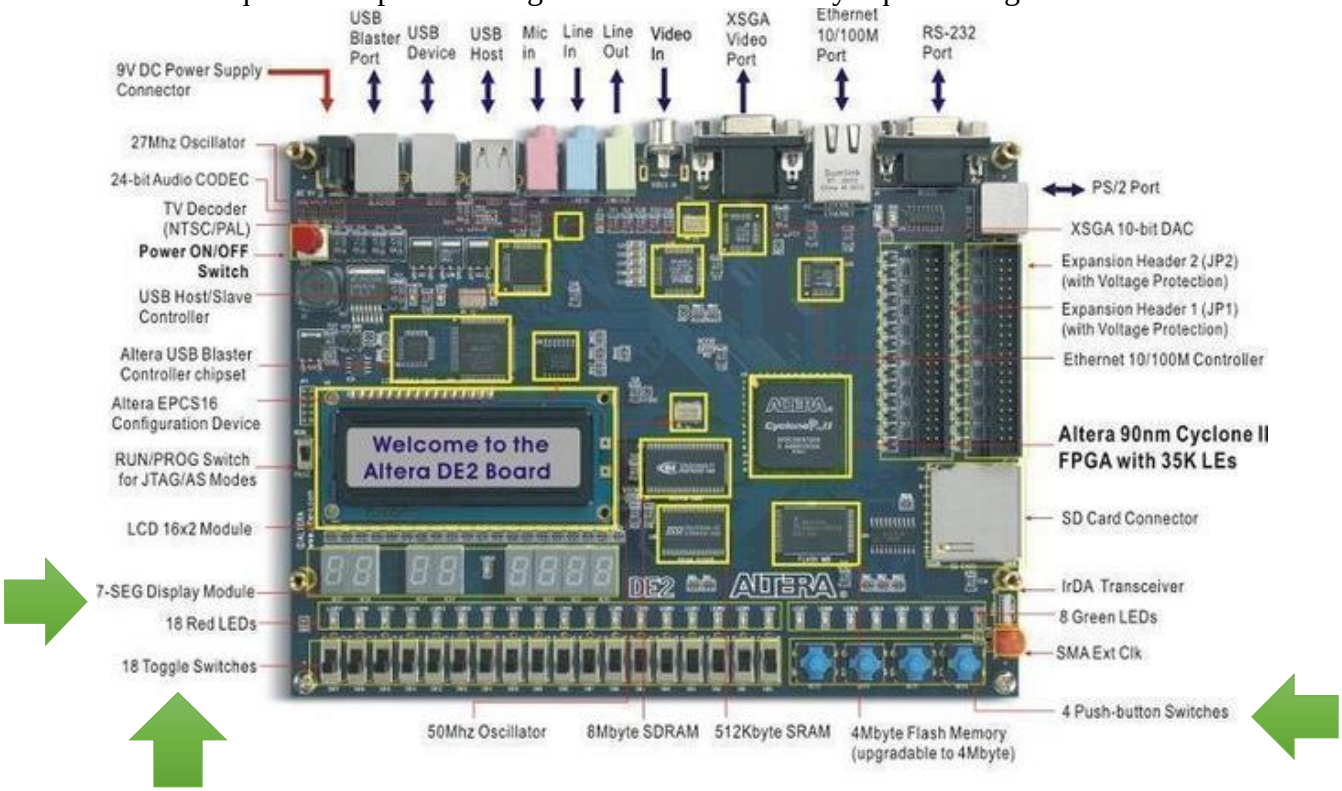
El diseño arquitectónico:

El sistema de cómputo a utilizar se fundamenta en la arquitectura Multiciclo (MIPS). Se modificaron los tipos de instrucciones y el tamaño de bits que estas ocupan. En la arquitectura se eliminaron algunas unidades que no eran necesaria para la ejecución de las instrucciones implementadas, todo lo anterior con el objetivo de optimizar la arquitectura de acuerdo con nuestras necesidades.

El diseño de interfaz:

Se pensaron cuatro opciones. La primera es conectar un teclado a la FPGA para el ingreso de los datos, la segunda es conectar un monitor por medio del puerto RS232 y usar los botones de la FPGA o en efecto la opción del teclado, lo cual aparentemente parece complicado. La tercera es utilizar el Arduino como interpretador de las señales (Interfaz) que arrojaría la FPGA y, por último, la cuarta y hasta ahora la más viable es utilizar los 4 botones de la FPGA para ingresar los datos y un interruptor para confirmar que el usuario ya ingresó todo. Para mostrar los resultados se utilizarán los displays 7 segmentos. De cualquier forma, se busca una interfaz sencilla dado que simplemente se van a mostrar las probabilidades resultantes de comparar los genotipos.

NOTA: El interruptor que funciona como una validación de que la persona ya ingresó los datos será una señal a la unidad de control para que muestre los datos y posteriormente los cargue en la memoria. Se pensó en crear un interruptor que emita una señal de reset para el caso en que una persona ingrese mal un dato y quiera ingresarlo de nuevo.



El diseño de la arquitectura:

Largo de memoria de instrucciones: Las direcciones de memoria cuentan con 9 bits, es decir 512 posibles direcciones. Este tamaño fue necesario porque al crear un tipo de instrucción especial para las instrucciones Sm, Lm, La, Jeq, Jst, el opcode queda con 3 bits, los registros Rs y Rd quedan con 5 bits cada uno y al final el campo de la constante o dirección queda con 9 bits. Aunque la Memoria de Datos es más pequeña (7 bits de ancho) los dos bits adicionales son porque el espacio de constante es tanto para inmediatos como para alojar direcciones de salto.

Ancho de memoria de instrucciones: Es de 22 bits debido a que esa cantidad requiere el tipo de instrucción que se manejará.

Largo de memoria de datos: El ancho de la memoria es de 7 bits puesto que el máximo dato a guardar es el número 100 que necesita 7 bits en conversión a binario.

En el diseño de la arquitectura algunas de las decisiones importantes fueron:

- **¿Cuántas memorias utilizar?**
  Se tomó la decisión de utilizar dos memorias, una para instrucciones y otra para datos con el objetivo de evitar errores en el almacenamiento y el manejo de la información.

- **¿Cómo definir los tipos de instrucciones?**
  Fue bastante complejo porque de definir bien los tipos dependía la estructura y complejidad de la arquitectura, la máquina de estados y el código en ensamblador.

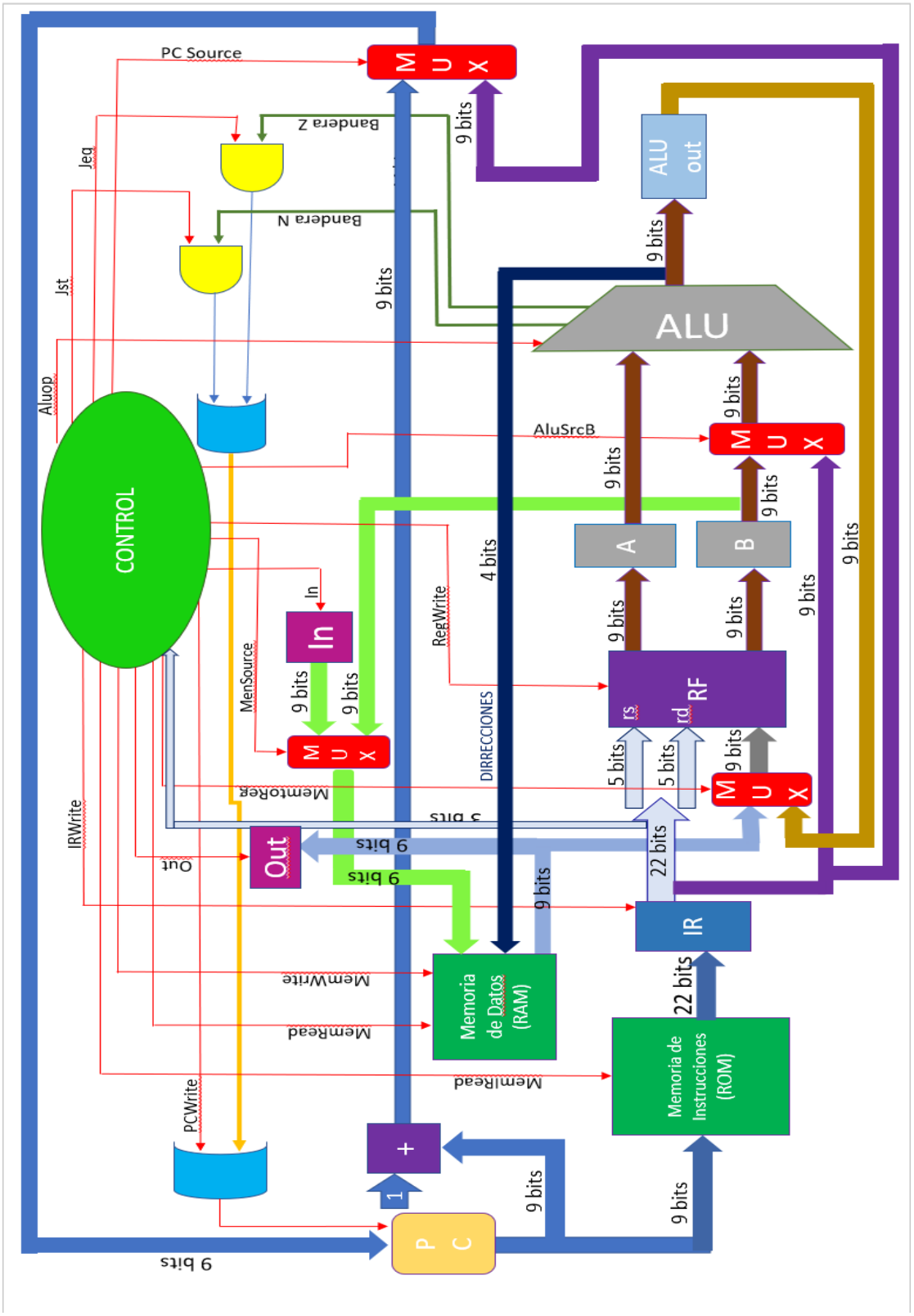
- **¿Cómo se definió la cantidad de registros a utilizar y el ancho de los registros?**
  En el código se usó más de 16 registros por eso fue necesario utilizar 5 bits para el largo de los registros y el ancho corresponde a la cantidad de bits de los datos.

- **¿Por qué no se creyó necesario el MDR?**
  No se necesitó el MDR porque el dato va a el Register File (RF) y se guarda en un registro en el mismo ciclo en el que se saca de memoria. Entonces el dato no se pierde.

En la implementación del código ensamblador algunas dificultades fueron:

- No fue fácil pasar de alto nivel a bajo nivel porque las comparaciones no fueron tan fáciles de implementar. Buscar la forma óptima de calcular las probabilidades de los cruces entre el padre y la madre fue otro problema a pensar. Otro problema fue pasar lo escrito por el teclado o botones a una codificación fácil de usar para el grupo y luego comparar a partir de ella.

Diagrama Estructural de la Arquitectura del procesador



Descripción y Formato de Instrucciones y Registros

Instruction format:

I type instructions	Opcode	Rd	Rd	Inmediate/Adress
# of bits	3	5	5	9

Data_Memory (Ram)	Data
# of bits (width)	9
# of bits (length)	16

Instruction_Memory (Rom)	Instruction
# of bits (width)	22
# of bits (length)	256

Instruction set:

- Sm (Store memory): Carga un dato en DataMemory.
- Lm (Load memory): Carga un dato de Datamemory en un registro.
- La (Load address): Carga una dirección de memoria en un registro.
- Jeq (Jump equal): Salta si los dos registros son iguales.
- Jst (Jump smaller than): Salta si el dato que contiene un registro es más pequeño que el otro.
- Addc (Add constant): Suma la constante con el dato que contiene un registro.
- Input: Carga un inmediato ingresado por el usuario a la memoria de datos (en una parte específica que recibe los valores de los genotipos).
- Output: Manda el dato que contiene un registro a mostrarse en los displays.

Ejemplos:

sm r16,(r10),000000000
lm r18,(r10),000000000
la r7,r0,00000000100
jeq r15,r17,000010101
jst r19,r21,101001011
addc r13,r14,000101000
input r10,r0,001000100
output r20,r0,100000001

Register and memory format:

- La memoria de instrucciones consta de 22 bits de ancho y 2^9 posiciones de largo.
- Los registros posibles para usar son 32.
- El RF consta de 2^5 posiciones y de 9 bits de ancho.