

Ejercicio 9a:

9 a. Tamaño de la memoria física = 2GB; Registros = 624 (2^9) de 64 bits divididos en 4 partes de 16 bits; Tamaño de la unidad direccionable: 1B
 Para diseñar un formato de instrucciones de longitud fija de 64 bits, nos primero que # unidades direccionables = $\frac{2GB}{1B} \cdot \frac{2^{30}B}{16B} = 2^{31} \Rightarrow$ la cantidad de bits de la dirección de memoria equivale a $\log_2(2^{31}) = 31$. A su vez, como hay 2^9 registros, los puedo distinguir con una cadena de $\log_2(2^9) = 9$ bits.
 Ahora veo que la instrucción MOV r, q es la que requiere la mayor reserva al tener que reservar en un caso 62 bits correspondientes a las direcciones de memoria. Esto deja 2 bits libres para el código de operación y el modo de direccionamiento. Como a los bits como 11 para indicar que la operación es MOV con modo de direccionamiento directo en ambos operandos. Véase que para los otros modos de direccionamiento el espacio a reservar es menor ya que el espacio reservado para una de las direcciones de memoria (al menos) pasará de 31 a 10 bits, por lo que tengo la combinación 01 para indicar que la instrucción es MOV y los operandos son dirección de memoria y registro o ambos registros. siendo que ahora tengo 21 bits disponibles para dos combinaciones y distinguir estos 3 casos, nos que puedo tomar sólo los primeros 2 para distinguirlos y dejar el resto en 0. Si el primer operando es una dirección toma 10, si es el segundo 01 y si ambos son registros 00. De esa forma, para MOV r, q tengo:

1	Dirección de memoria (31 bits)	1	Dirección de memoria (31 bits)	→	MOV [p], [q]	Notese que usamos 0 para indicar que una cantidad con			
0	Dirección de memoria (21 bits)	1	10	0 (19 bits)	Registro (10 bits)	reservar de bits es nulo (todos ceros). Otro para el resto			
0	Dirección de memoria (31 bits)	1	01	0 (19 bits)	Registro (10 bits)				
0	0 (5 bits)	Registro (10 bits)	0	0 (6 bits)	Registro (10 bits)	1	0 (5 bits)	→	MOV R _p , R _q

Los que luego tengo dos instrucciones (INT R_u(i), R_v, R_w y SDV R_u, R_v, R_w(i)) que tienen que reservar 32 bits provenientes de los números de los 3 registros y una parte del primero o el tercero según la instrucción. Para esto reservo los bits 32-61 para los números de registro y dejo los bits 31 y 63 en 0 para distinguir a otras operaciones de MOV (contando de 0 a 63 desde el menor significado bit). Esto deja al bit 62 disponible para distinguir una instrucción de la otra, y de los 31 bits menos significativos reservo los bits 30 y 29 para escribir el número de parte del registro, dejando al resto en caso:

0	0	Registro (10 bits)	Registro (10 bits)	Registro (10 bits)	0	Parte del Registro (2 bits)	0	(24 bits)	→	INT R _u (i), R _v , R _w
0	1	Registro (10 bits)	Registro (10 bits)	Registro (10 bits)	0	Parte del Registro (2 bits)	0	(24 bits)	→	SDV R _u , R _v , R _w (i)

Requiere a esto tengo las instrucciones VEC y ADD que toman 3 registros para lo cual debo reservar 30 bits. Para distinguirlos de los anteriores tomamos los bits 31 y 63 en 0 (no es MOV) y el bit 28 en 1 (no es ninguna de las 2 anteriores). Luego, se usaron los bits 32-61 para los registros y el bit 62 para distinguir entre una y la otra. El resto quedarán en 0:

0	0	0	Registro (10 bits)	Registro (10 bits)	Registro (10 bits)	0	00	1	0 (28 bits)	→	VEC R _u , R _v , R _w
0	1	0	Registro (10 bits)	Registro (10 bits)	Registro (10 bits)	0	00	1	0 (28 bits)	→	ADD R _u , R _v , R _w

Por último tomamos la instrucción CLR R_u(i) para la cual debemos reservar 12 bits correspondientes al número de registro y la parte del mismo. Para distinguir de los anteriores tomaremos los bits 31, 62 y 63 en caso, los bits 28-30 en caso y el bit 27 en 1. Luego usaremos los bits 52-61 para el número de registro y los 2 anteriores para la parte (el resto en 0), dejando:

0	0	0	Registro (10 bits)	Parte i (2 bits)	0 (16 bits)	00	00	1	0 (27 bits)	→	CLR R _u (i)
---	---	---	--------------------	------------------	-------------	----	----	---	-------------	---	------------------------